BEGV643A
USER MANUAL

LCD Embedded System,
Atmel ATmega644p MCU,
Graphic 320x240 STN LCD,
LED/white backlight,
RS232/RS422/RS485, I^2C /SPI
64KB in-system programmable Flash
No Operating System required

CONTENTS

Precaution

FCC

WARNING

CAUTION

This device is designed to meet the requirement in part 15 of the FCC rules. Operation is subject to conditions ruled under FCC part 15.

Please check packing content upon receiving BEGV643A parcel, make sure that all materials and options are packed inside parcel according to your order.

Packing Contents Check-List

- □ BEGV643A LCD Embedded module
- □ Touch panel
- □ Software Utility Disc
- □ ISP Cable (option)
- □ ATMEL software development tool

Chapter 1 Introduction

Abstract

This chapter is to offer you basic information regarding BEGV643A, to help you incorporate BEGV643A into your system.

Contents include:

1-1 Features

1-2 Board Layout

1-3 Block Diagram

1-4 Mechanical Dimension

1-5 Board Specifications

1-6 Ordering information

1-1 Features

This BEGV643A is designed based on ATmega644p microprocessor, which requires no operating system to run on. Together with a 320x240 STN LCD and LED backlight built-in, this all-in-one LCD embedded system BEGV643A help designer enhance a compact design with cost saving, space saving, and design phase saving.

Armed with RS232, RS422/485, I^2C and SPI interface port, this BEGV643A communicates many devices and peripherals. The BEGV643A is therefore suitable to sit as a industrial control panel for factory automation equipment, electronics instrument, HMI (human-machine interface), office automation equipment, medical equipment, parking system, ticketing system.. and so on.

There are five LCD colors among choices: STN/gray, STN/yellow-green, STN/blue, FSTN/Positive, and FSTN/negative. All comes with LED/white backlight.

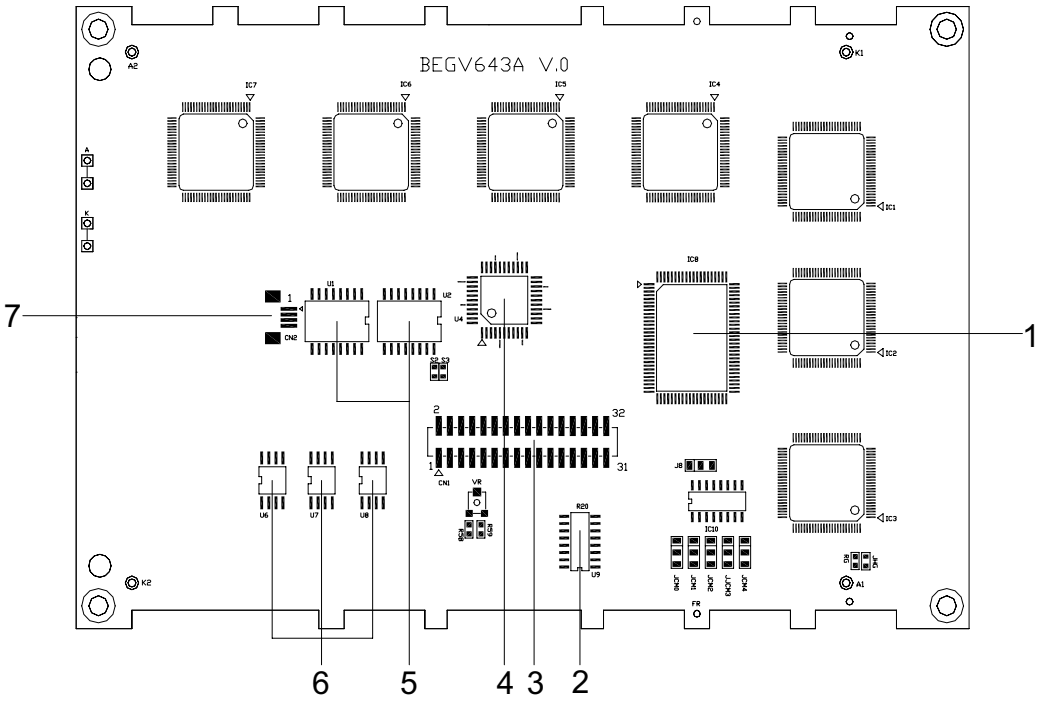
64KB in-system self-programmable Flash offers sufficient ROM size for designer to develop software, further to control LCD, touch panel, and interfaces.

Designer may simply design this BEGV643A into your application as you are designing a ATMEL CPU board, without worrying LCD module and other interfaces, since they are all on one board.

1-2 Board Layout

This layout shows the location of each important IC, connector and jumper. Please refer to chapter 2 for further information on jumper and connector.

(Drawing 1.2)



1 :LCD controller **7** : CN2

2 :RS232

3: CN1

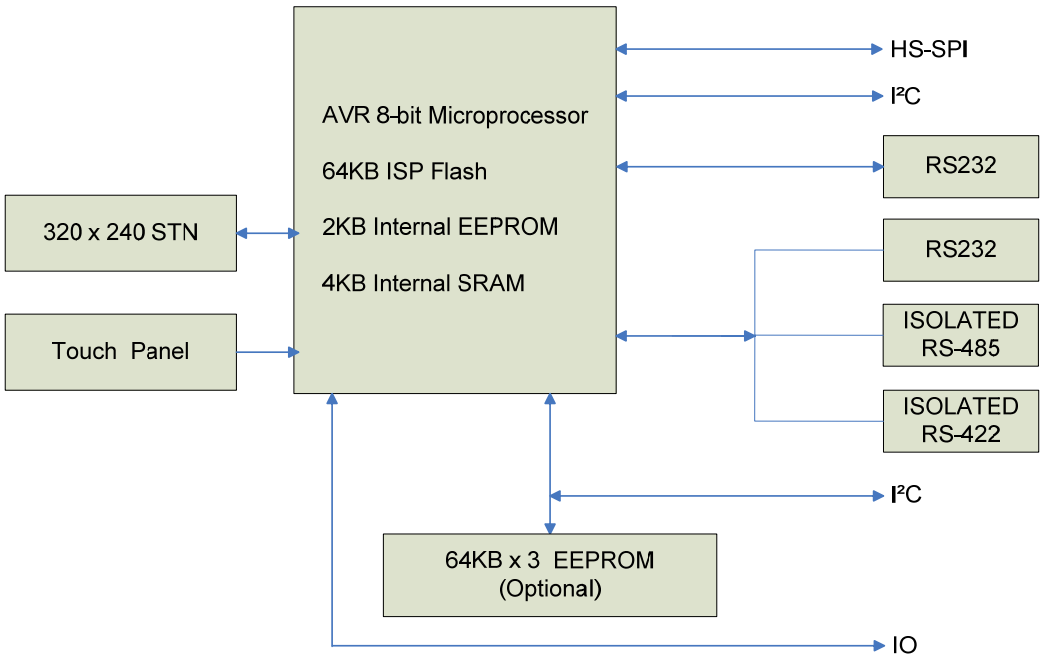
4 :Microprocessor

5 :RS-485/422

6 :EEPROM

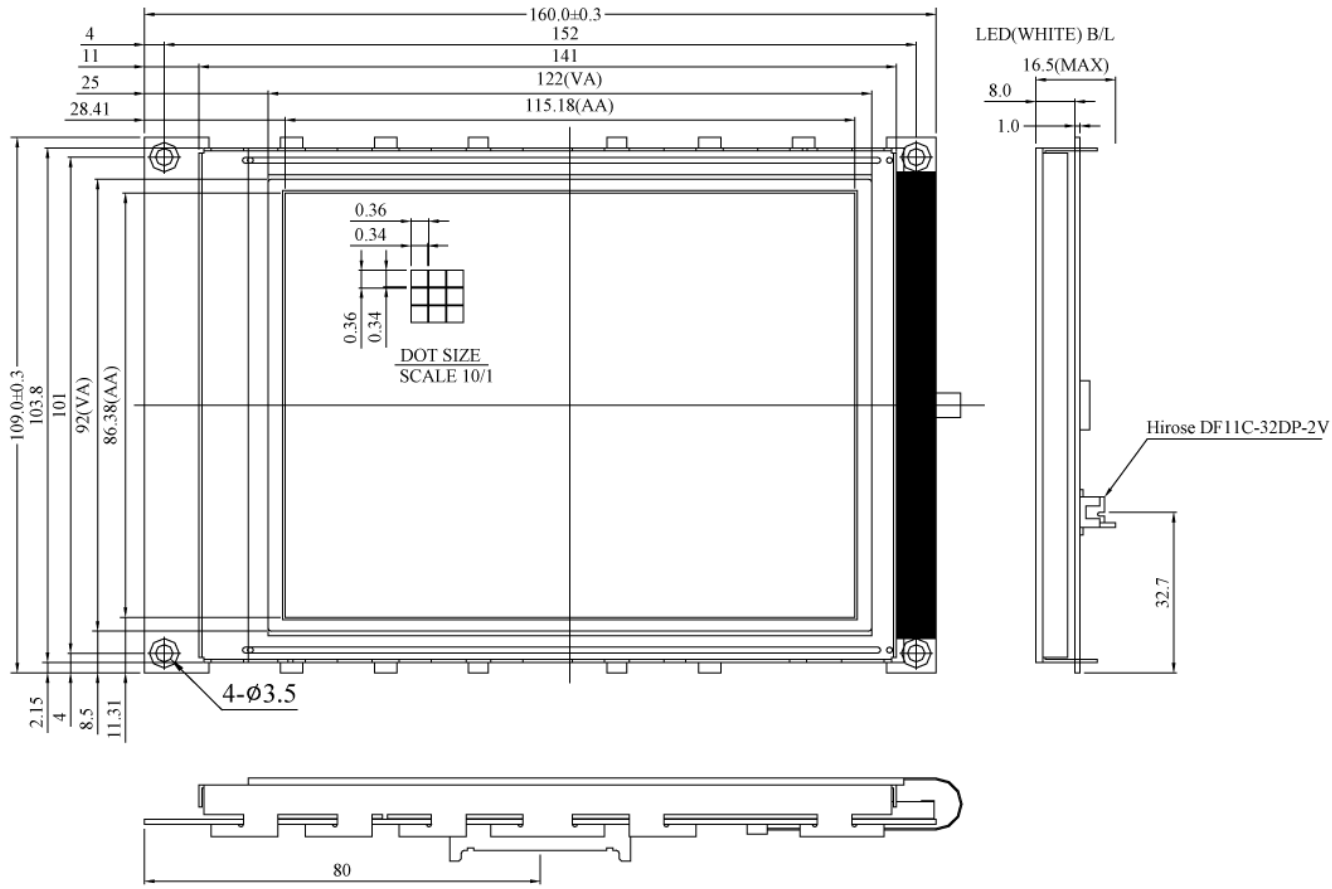
1-3 Block Diagram

(Drawing 1.3)



1-4 Mechanical Dimension

(Drawing 1.4)



1-5 Board Specifications

(Table 1.5)

MCU	High-performance, Low-power AVR® 8-bit microprocessor ATMEL ATmega644P
Memory	64K Bytes In-System Self-Programmable Flash 2K Bytes Internal EEPROM 4K Bytes Internal SRAM 3 x 64K Bytes External EEPROM(optional)
Display	Support 8-bit single-scan resolution 320 x 240 monochrome STN LCD, with edge LED white backlight only
Touch Panel(optional)	Support four-wired resistive touch panel
Serial Ports	Support 1 x RS232 port, and 1 x RS232/RS422(isolated)/RS485(isolated) co-shared port Support 1 x full-duplex, three-wired synchronous data transfer SPI port Support 1 x two-wired serial interface to 250 KHz data transfer speed

1-6 Ordering Information

(Table 1.6)

Part No.	Description	RS232-A	RS232-B	RS422	RS485
BEGV643A	Dual RS232	☆	☆		
BEGV643A1	One RS232	☆			
BEGV643A2	One RS232/One RS422	☆		☆	
BEGV643A3	One RS232/One RS485	☆			☆

Display: FSTN/Positive LCD,LED/White Backlight(Default)

Chapter 2 Installation

Abstract

This chapter is to offer designer fundamental information of BEGV643A jumpers and connectors, in order to help designer configure correct setting and connection between BEGV643A and system application.

Contents include:

2-1 Jumpers

2-2 Connectors

2-1 Jumpers

This section is to indicate location and function of each jumper on BEGV643A, which user can arrange according to the needs of different application desired. Be careful when setting jumper, user maybe need tool such as needle-nose pliers to help setting. Please note, jumpers not described here are intended to keep as factory default setting. Please consult Bolymin before trying to change default setting.

The table listed below describes location and function of each available jumper.

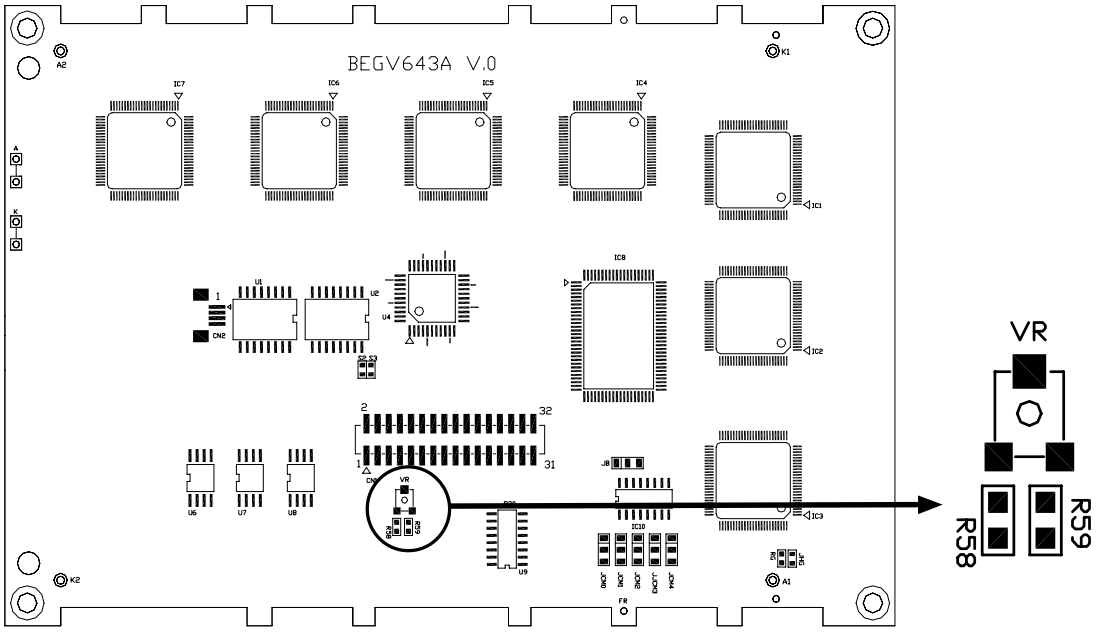
Jumpers: (Table 2.1)

Label	Function
VR	LCD contrast control
RG	Frame ground
S2,S3	RS-422/RS-485 VDD/ground
JHG	Screw hole ground

Detail location and function of each jumper is illustrated below.

2-1-1 Contrast Adjust

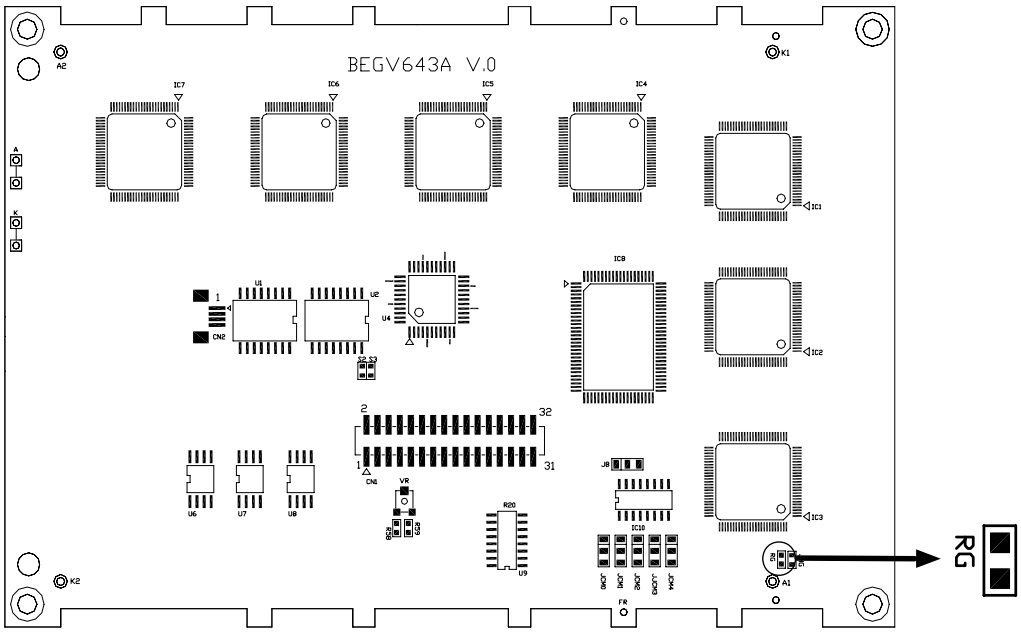
(Drawing 2.1.1)



VR	Contrast adjust	
10K ohm	Inside	default
N/A	Outside	

2-1-2 Frame Ground

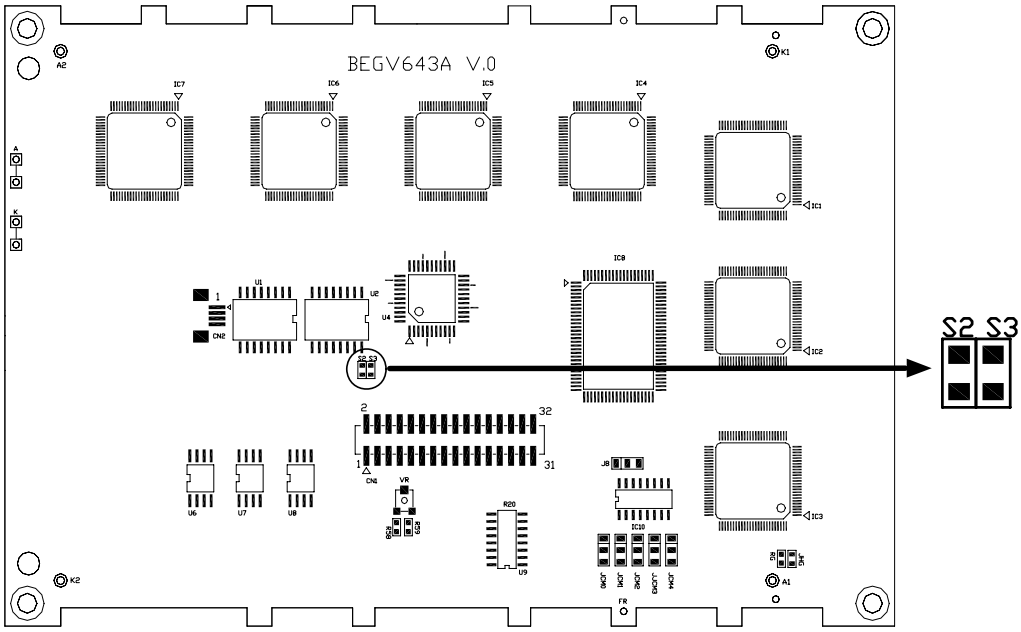
(Drawing 2.1.2)



RG	Frame ground	
short	Connect metal frame with GND	
open	Not connected metal frame with ground	default

2-1-3 RS-422/RS-485 VDD/Ground

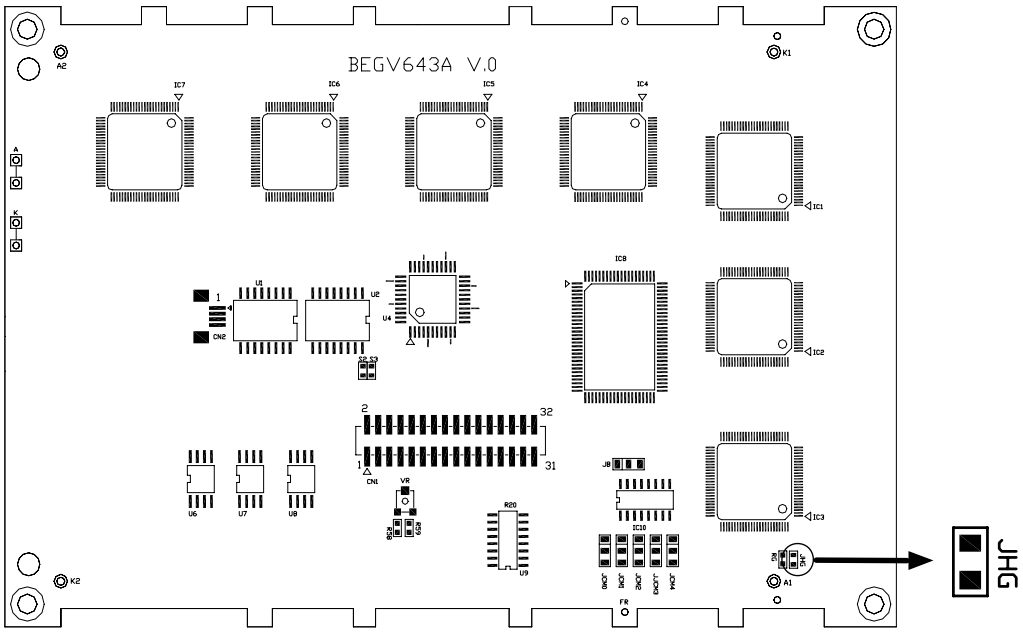
(Drawing 2.1.3)



S2	S3	RS-422/RS-485 VDD/Ground	
short	short	internal VDD/ground for isolated RS-422/485	
open	open	External VDD/ground for isolated RS-422/485	default

2-1-4 Screw hole ground

(Drawing 2.1.4)



JHG	Frame ground	
short	Connect screw hole with GND	
open	Not connected screw hole with ground	default

2-2 Connectors

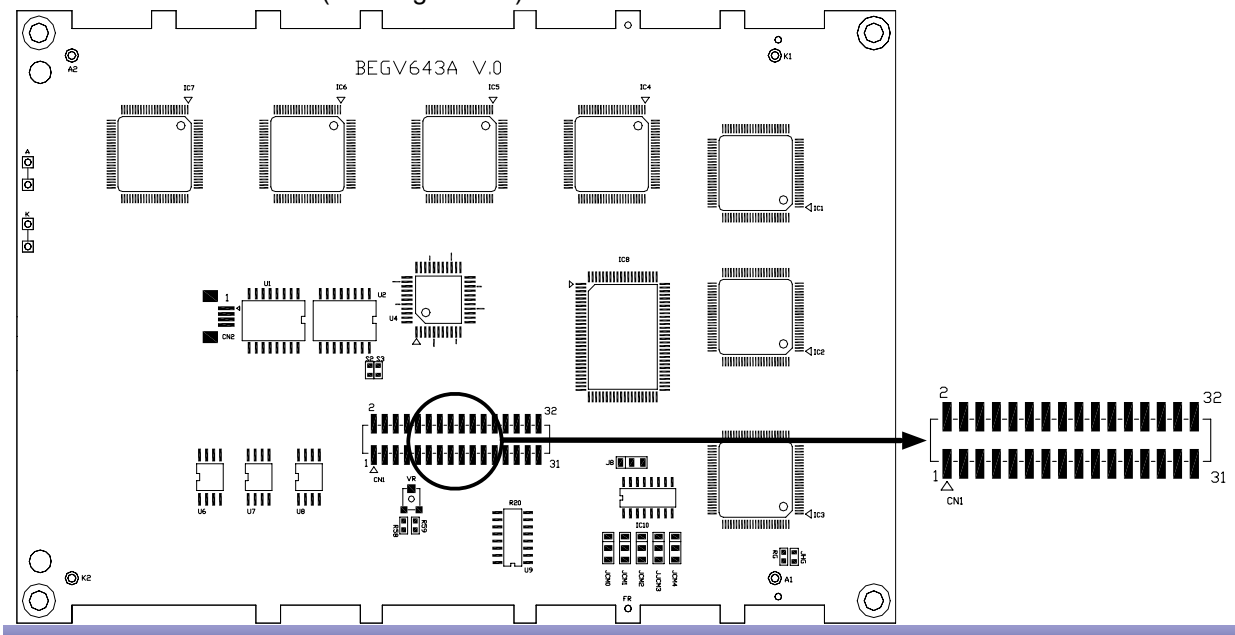
Connectors are the key link between BEGV643A and external devices. Detail locations and functions of available connectors are tabled and illustrated below.

Connectors: (Table 2.2)

Label	**Pin No.**	**Function**
CN1	32	Connector

2-2-1 Connectors & Pin Definition

- **Connector CN1** (Drawing 2.2cn1)



Pin Types	
I	=Input
O	=Output
Bi	=Input / Output (Bi-Directional)
U	=User defined
P	=Power

- **Pin Definition: BEGV643A-Dual RS232** (Table 2.2.1a)

Pin No.	Signal	Pin No.	Signal
1	GND	2	EEPSDA
3	VDD	4	EEPSCL
5	NC	6	EEPWP
7	NC	8	IOB
9	/Reset	10	NC
11	/SS	12	NC
13	MOSI	14	NC
15	MISO	16	NC
17	SCK	18	NC
19	RX0	20	NC
21	TX0	22	NC
23	RX1	24	NC
25	TX1	26	NC
27	SDA	28	NC
29	SCL	30	NC
31	IOA	32	NC

- **Pin Definition: BEGV643A1-One RS232** (Table 2.2.1b)

Pin No.	Signal	Pin No.	Signal
1	GND	2	EEPSDA
3	VDD	4	EEPSCL
5	NC	6	EEPWP
7	NC	8	IOB
9	/Reset	10	NC
11	/SS	12	NC
13	MOSI	14	NC
15	MISO	16	NC
17	SCK	18	NC
19	RX0	20	NC
21	TX0	22	NC
23	NC	24	NC
25	NC	26	NC
27	SDA	28	NC
29	SCL	30	NC
31	IOA	32	NC

.

- **Pin Definition: BEGV643A2-One RS232/One RS422** (Table 2.2.1c)

Pin No.	Signal	Pin No.	Signal
1	GND	2	EEPSDA
3	VDD	4	EEPSCL
5	NC	6	EEPWP
7	NC	8	IOB
9	/Reset	10	NC
11	/SS	12	422RP
13	MOSI	14	422RN
15	MISO	16	422TP
17	SCK	18	422TN
19	RX0	20	VDD2
21	TX0	22	VDD2
23	NC	24	GND2
25	NC	26	GND2
27	SDA	28	NC
29	SCL	30	NC
31	IOA	32	NC

- **Pin Definition: BEGV643A3-One RS232/One RS485** (Table 2.2.1d)

Pin No.	Signal	Pin No.	Signal
1	GND	2	EEPSDA
3	VDD	4	EEPSCL
5	NC	6	EEPWP
7	NC	8	IOB
9	/Reset	10	NC
11	/SS	12	NC
13	MOSI	14	NC
15	MISO	16	485P
17	SCK	18	485N
19	RX0	20	VDD2
21	TX0	22	VDD2
23	NC	24	GND2
25	NC	26	GND2
27	SDA	28	EN485
29	SCL	30	NC
31	IOA	32	NC

2-2-1-1 Power & Ground (Table 2.2.1.1)

Signal	Type	Pin No.	Description
GND	P	1	Logic power supply (ground)
VDD	P	3	Logic power supply (+5V)
VDD2	P	20	External Power for isolated RS-422/485(+5V)
VDD2	P	22	External Power for isolated RS-422/485(+5V)
GND2	P	24	External Ground for isolated RS-422/485(ground2)
GND2	P	26	External Ground for isolated RS-422/485(ground2)

2-2-1-2 Serial I/O (Table 2.2.1.2)

Signal	Type	Pin No.	Description
/SS	U	11	SPI Chip select
MOSI	U	13	MOSI is the mater data output line, when SPI on module is configured as a master. When SPI is configured as a slave. This pin reverse the role.
MISO	U	15	MISO is the master data input line, when SPI is configured master. When SPI Is configured as a slave. This pin reverse the role.
SCK	U	17	SPI clock
RX0	I	19	Receiver of first RS232 with driver
TX0	O	21	Transmitter of first RS232 with driver
RX1	I	23	Receiver of second RS232 with driver
TX1	O	25	Transmitter of second RS232 with driver
SDA	U	27	Data of 2-wire serial interface, it can be programmed as IO.
SCL	U	29	Clock of 2-wire serial interface, it can be programmed as IO.
EEPSDA	Bi	2	Data of 2-wire serial interface for additional EEPROM update.
EEPSCL	Bi	4	Clock of 2-wire serial interface for additional EEPROM update
EEPWP	I	6	Write protect of additional EEPROM
422RP	I	12	no inverting receiver of RS422
422RN	I	14	inverting receiver of RS422
422TP/485P	Bi	16	When it configured as RS422 it act as no inverting transmitter, When is configured as RS 485 it acts as positive differential IO.
422TN/485N	Bi	18	When it configured as RS422 it acts as inverting transmitter, When is configured as RS 485 it acts as negative differential IO.
EN485	O	28	Enable RS-485

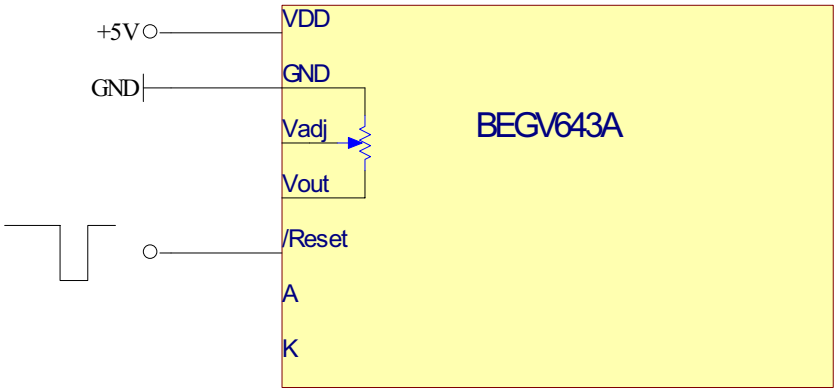
2-2-1-3 General I/O (Table 2.21.3)

Signal	Type	Pin No.	Description
/Reset	I	5	Auxiliary moment reset for external input
IOA	U	31	I/O port (ATmega644P portA.4)
IOB	U	8	I/O port (ATmega644P portA.7)

2-2-2 Pin vs. Function Diagram

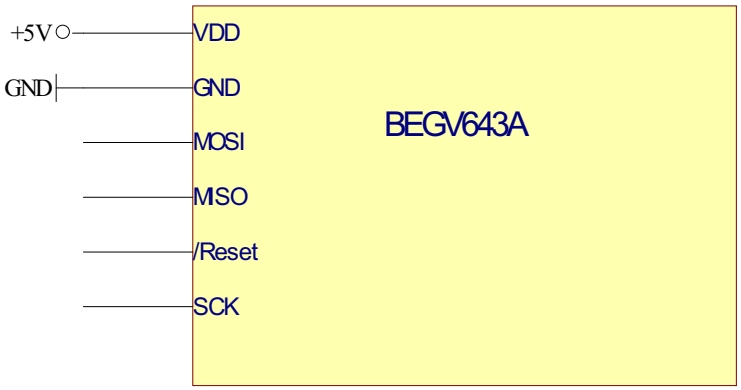
2-2-2-1 Power/LCD/Backlight

Diagram of system power supply, and contrast adjustment.



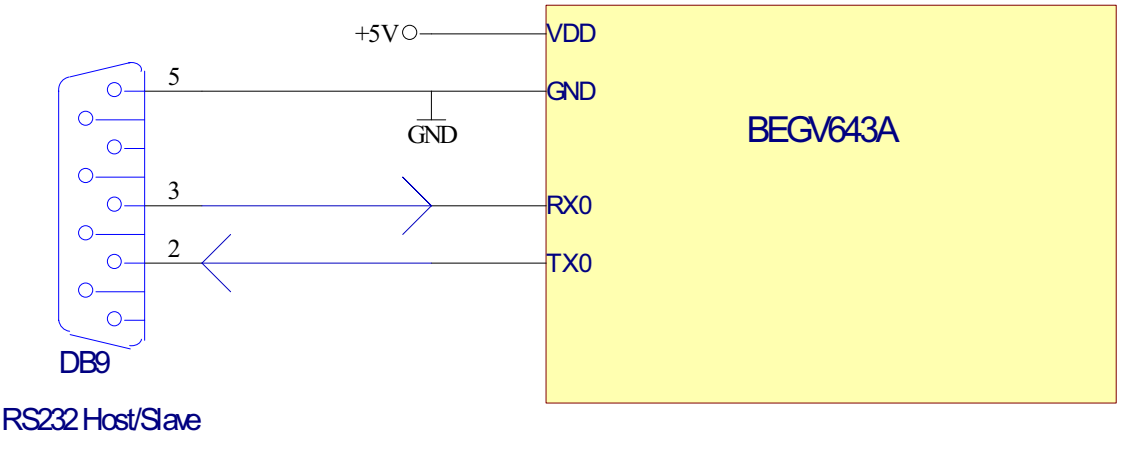
2-2-2-2 In-System Programming

BEGV643A offers in-system software burn-in function, which allows designer to write software into ATmega644p via ATMEL AVR ISP MKII writer without taking ATmega644p IC off board.
Please visit following website for more information on AVR ISP MKII writer.
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3808



2-2-2-3 RS-232

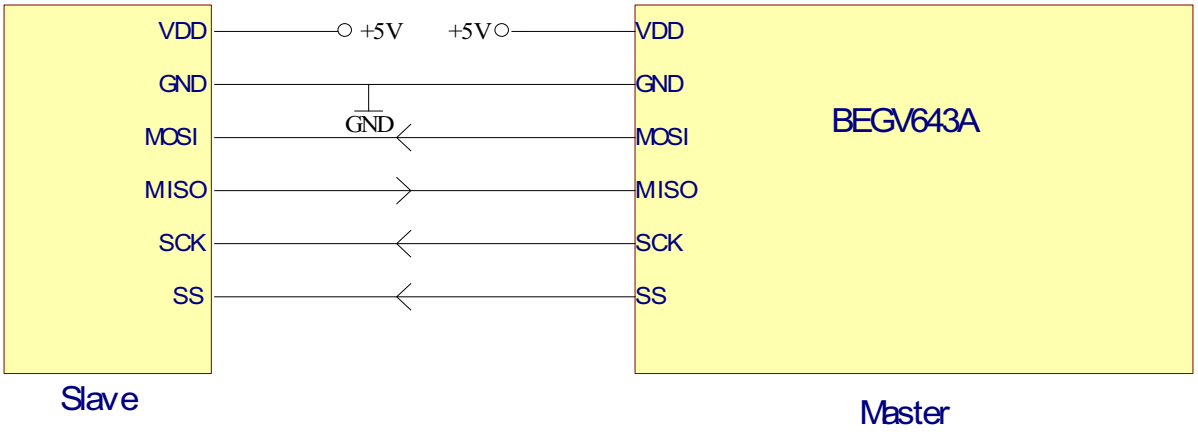
BEGV643A offers RS-232 port to contact PC or other RS232 device directly without ICL232.

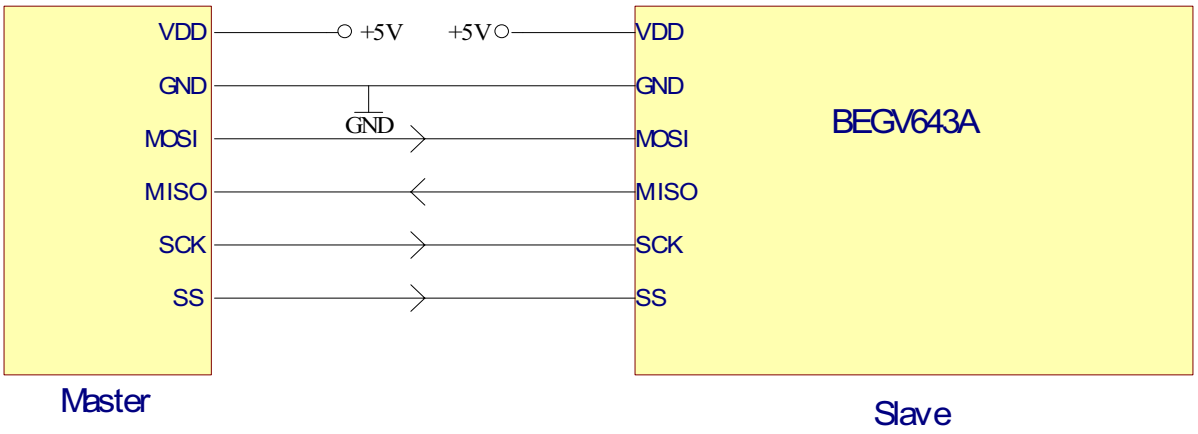


RS232 Host/Slave

2-2-2-4 SPI

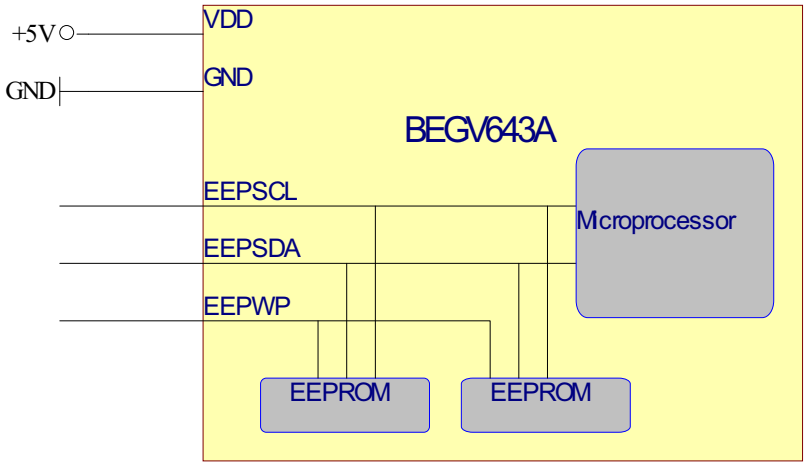
BEGV643A offers SPI port. Designer has to define this port as SPI by software, or to use Bolymin SPI driver (free utility).





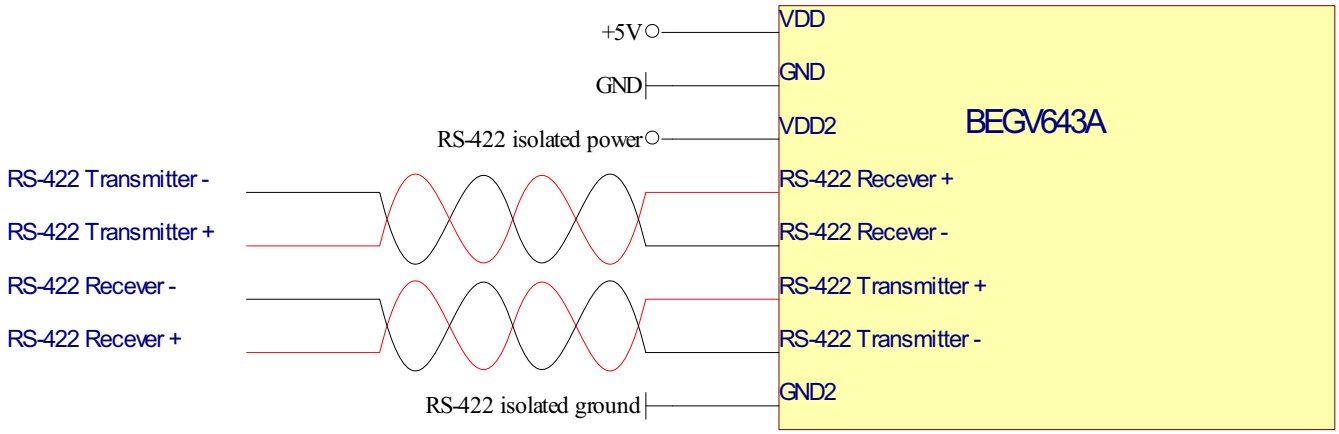
2-2-2-5 I^{2}C/EEPROM

BEGV643A offers I^{2}C port. Via this I^{2}C port, designer may control 64Kbytes x 3 EEPROM in-system and external I^{2}C devices.



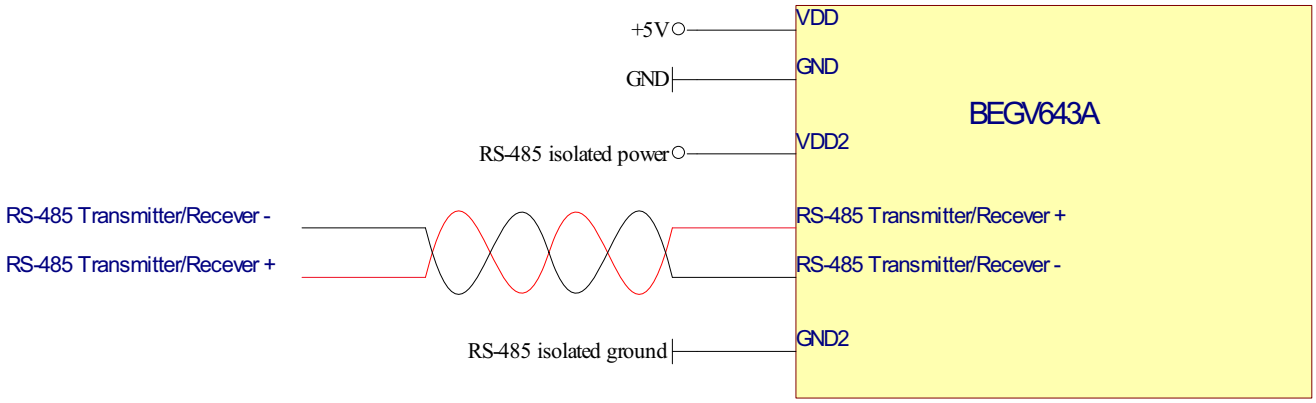
2-2-2-6 RS-422

BEGV643A offers 1 x RS-422(isolated) port.



2-2-2-7 RS-485

BEGV643A offers 1 x RS-485(isolated) port.



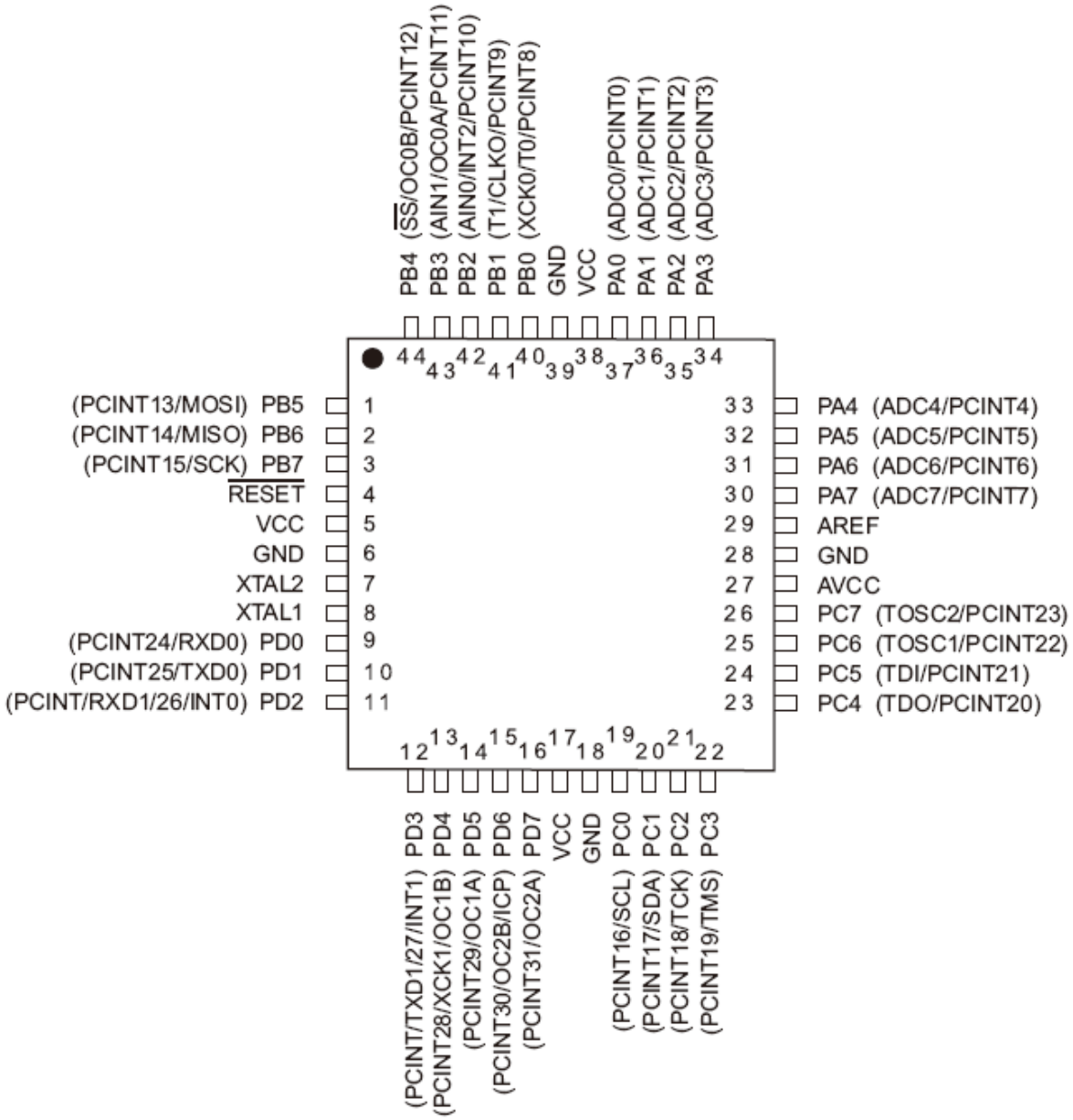
Chapter 3 MCU port mapping

Abstract

This chapter explains ATmega644p MCU pin configuration and port mapping toward key elements such as LCD, Touch Panel, RS-232, RS-422, RS-485, LED Backlight, EEPROM/I^2C, 2-wire serial port, SPI, and General I/O.

3-1 MCU Pin Configuration

(Drawing 3.1, ATmega644p MCU)



3-2 MCU Port Mapping

3-2-1 LCD Controller

(Table 3.2a)

MCU ATmega644P	LCD Controller
PORTB.0	DB0(LSB)
PORTB.1	DB1
PORTB.2	DB2
PORTB.3	DB3
PORTD.4	DB4
PORTD.5	DB5
PORTD.6	DB6
PORTD.7	DB7(MSB)
PORTC.7	LCD /RESET
PORTC.5	LCD RD
PORTC.4	LCD WR
PORTC.6	LCD C/D

3-2-2 Touch Panel

(Table 3.2b)

MCU ATmega644P	Touch Panel
PORTA.0	X1
PORTA.1	Y1
PORTA.2	X2
PORTA.3	Y2

3-2-3 RS-232/RS-422/RS-485

(Table 3.2c)

MCU ATmega644P	RS-232/422/485
PORTD.0	RX0
PORTD.1	TX0
PORTD.2	RX1
PORTD.3	TX1

(Table 3.2d)

MCU ATmega644P	RS-485
PORTA.5	Enable RS-485

3-2-4 Enable Backlight

(Table 3.2e)

MCU ATmega644P	LED backlight
PORTA.6	Enable Backlight

3-2-5 EEPROM/I^{2}C

(Table 3.2f)

MCU ATmega644P	EEPROM/ I^{2}C
PORTC.2	EEPROM SDA
PORTC.3	EEPROM SCL

3-2-6 2-wire serial port

(Table 3.2g)

MCU ATmega644P	2-wire serial port
PORTC.1	SDA
PORTC.0	SCL

3-2-7 SPI

(Table 3.2h)

MCU ATmega644P	SPI
PORTB.4	/SS
PORTB.5	MOSI
PORTB.6	MISO
PORTB.7	SCK

3-2-8 General I/O

(Table 3.2i)

MCU ATmega644P	General I/O
PORTA.4	IOA
PORTA.7	IOB

Chapter 4 Software Development Tool & Utility

Abstract

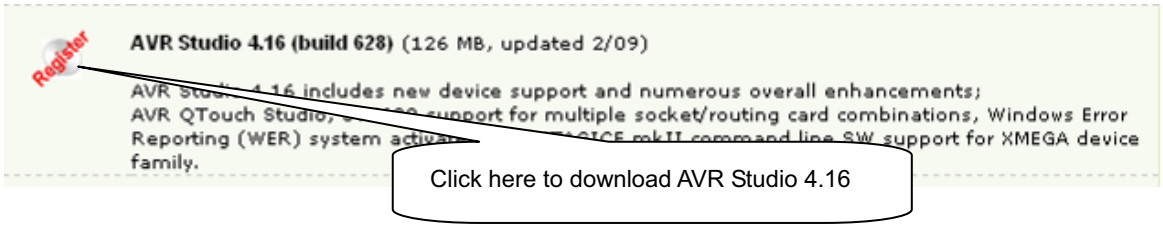
This chapter explains ATmega644p MCU software development tool AVR ISP mkII writer and Bolymin free software utilities.

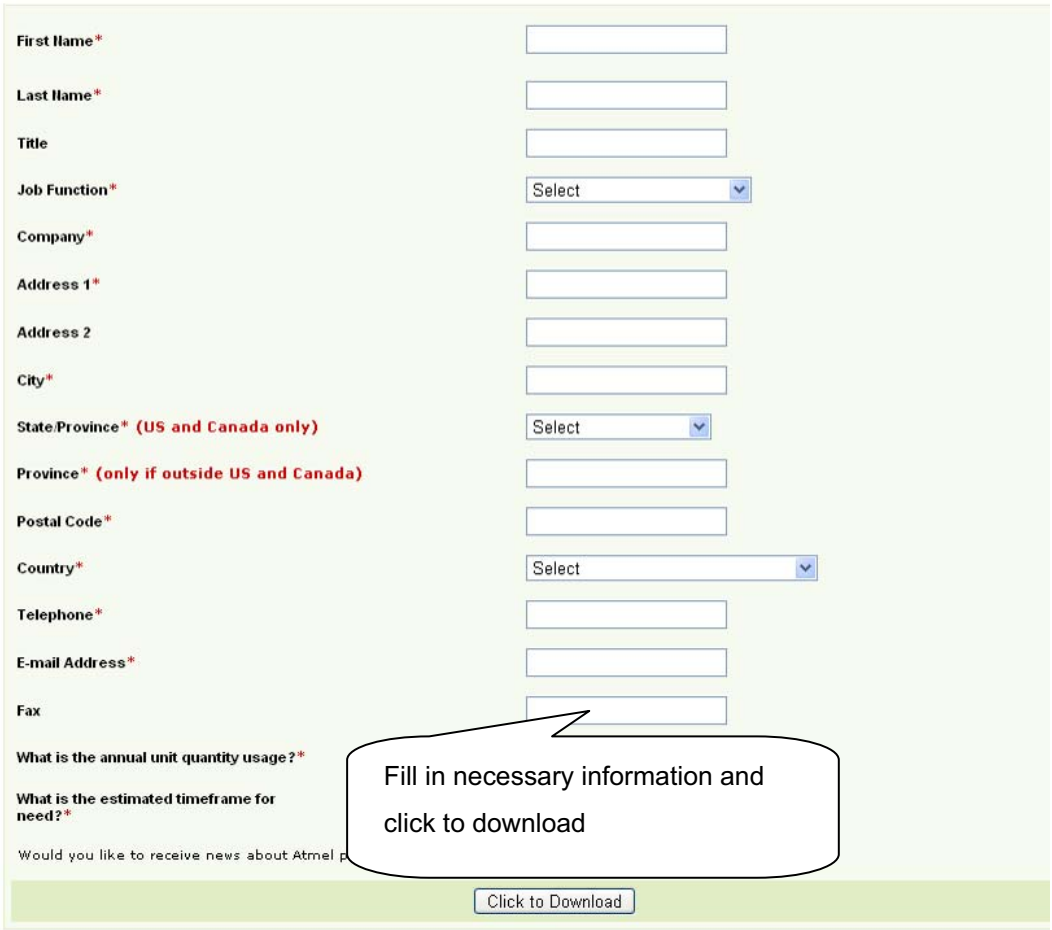
4-1 ATMEL ATmega644p Software Development Tool

Designers may download software development tool from AVR Studio website http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725, or from BOLYMIN utility disk.

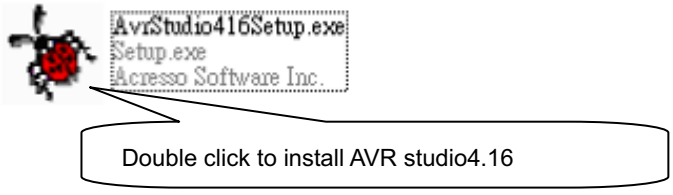
4-1-1 Download software from AVR Studio website

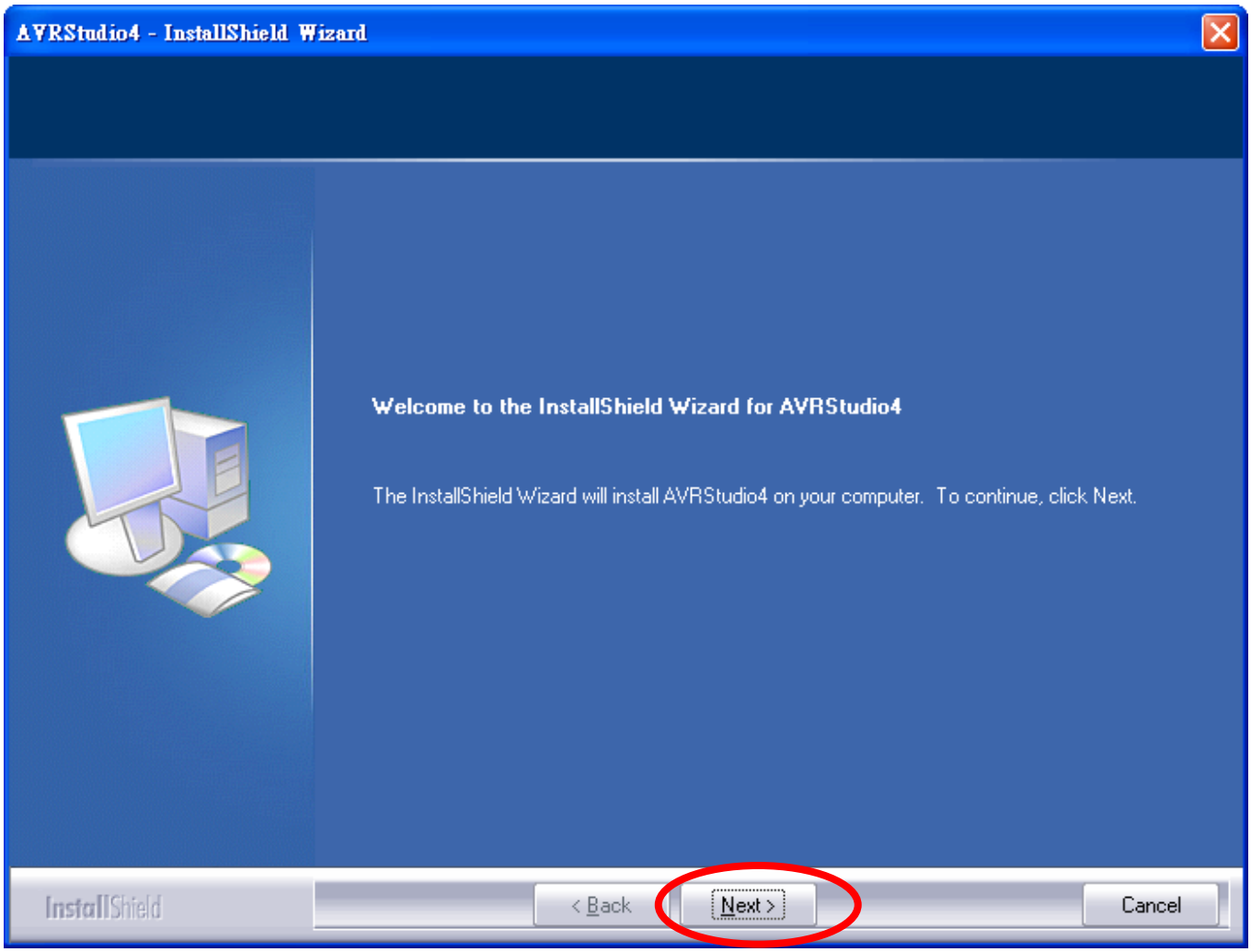
step1: Download design software

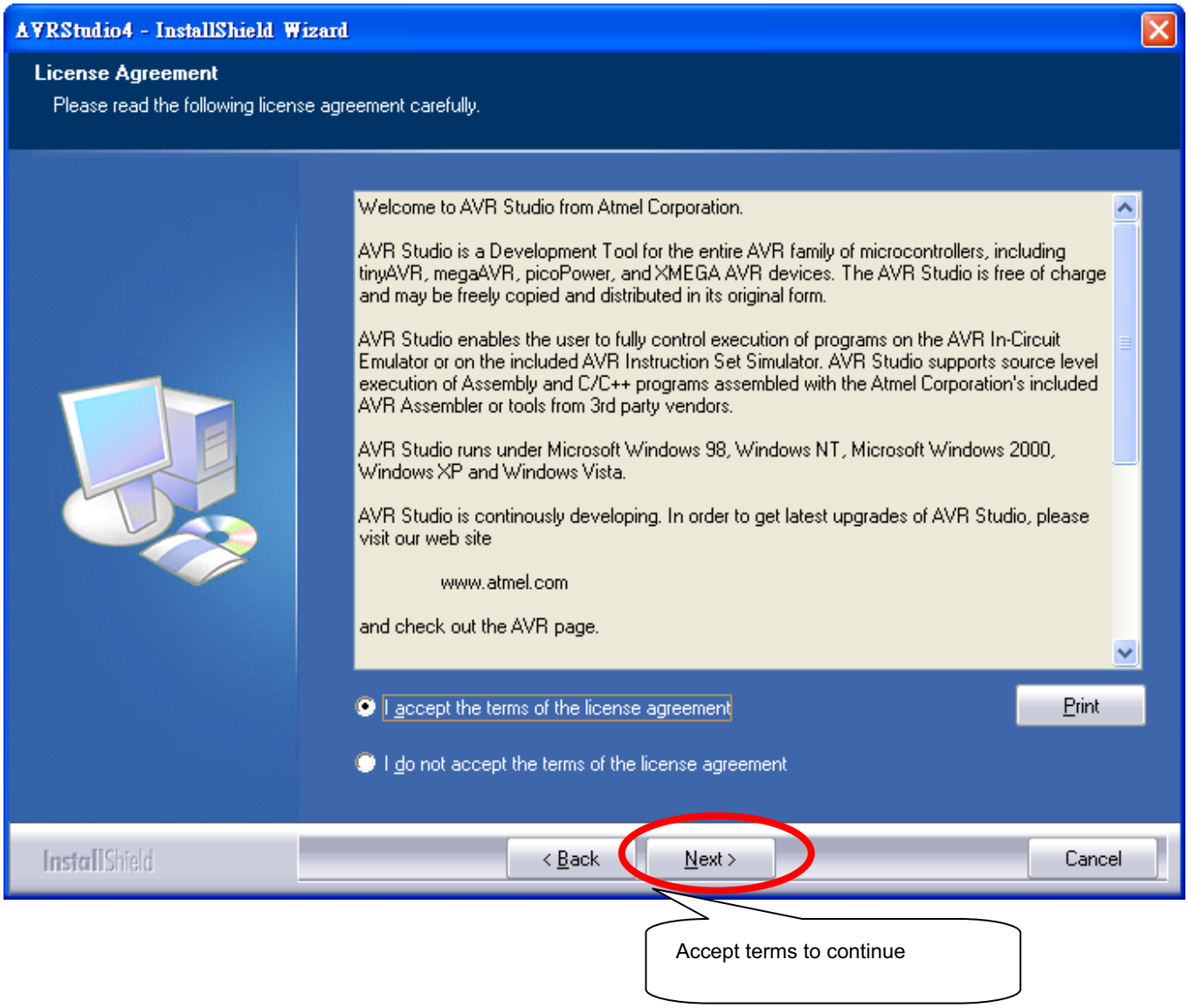


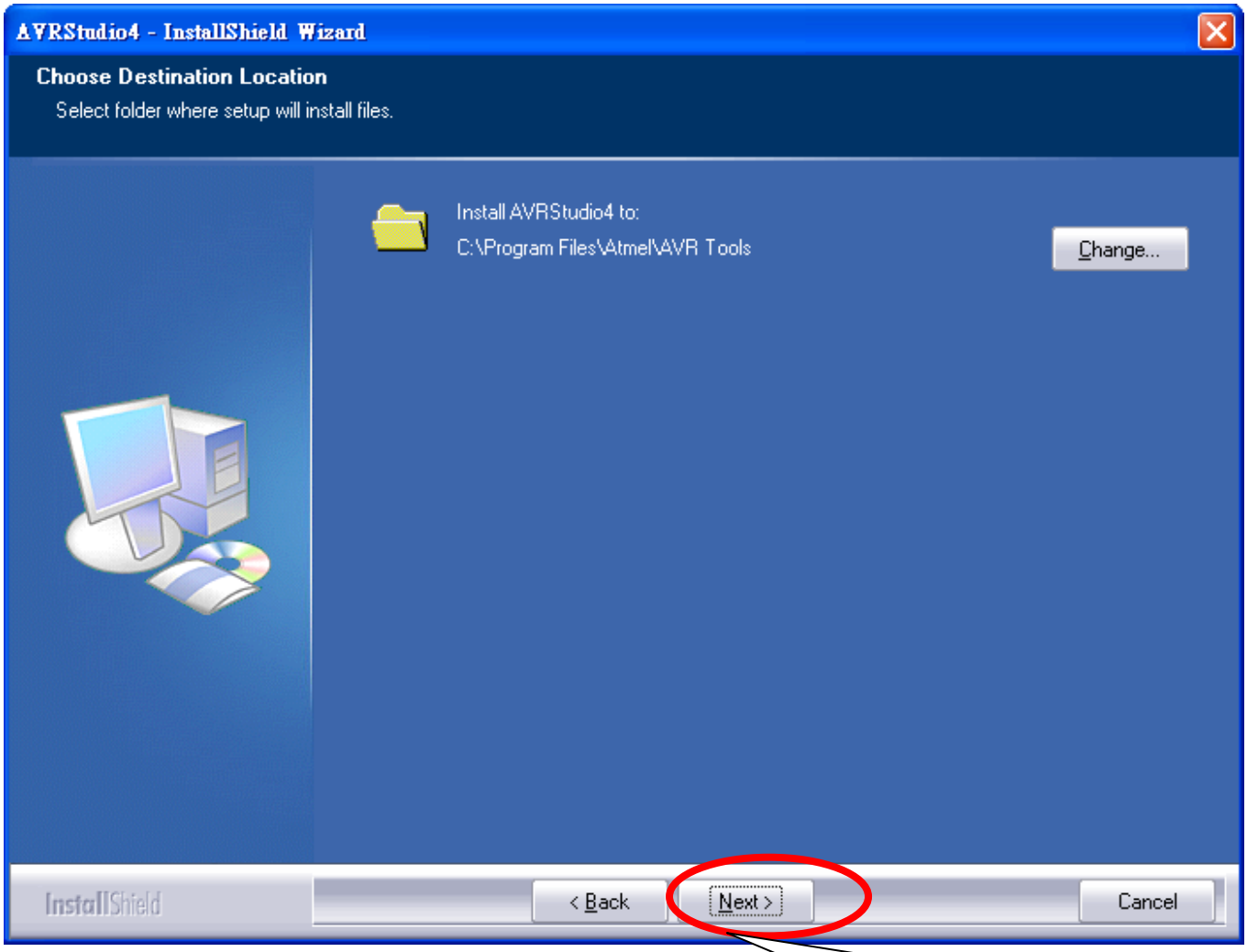


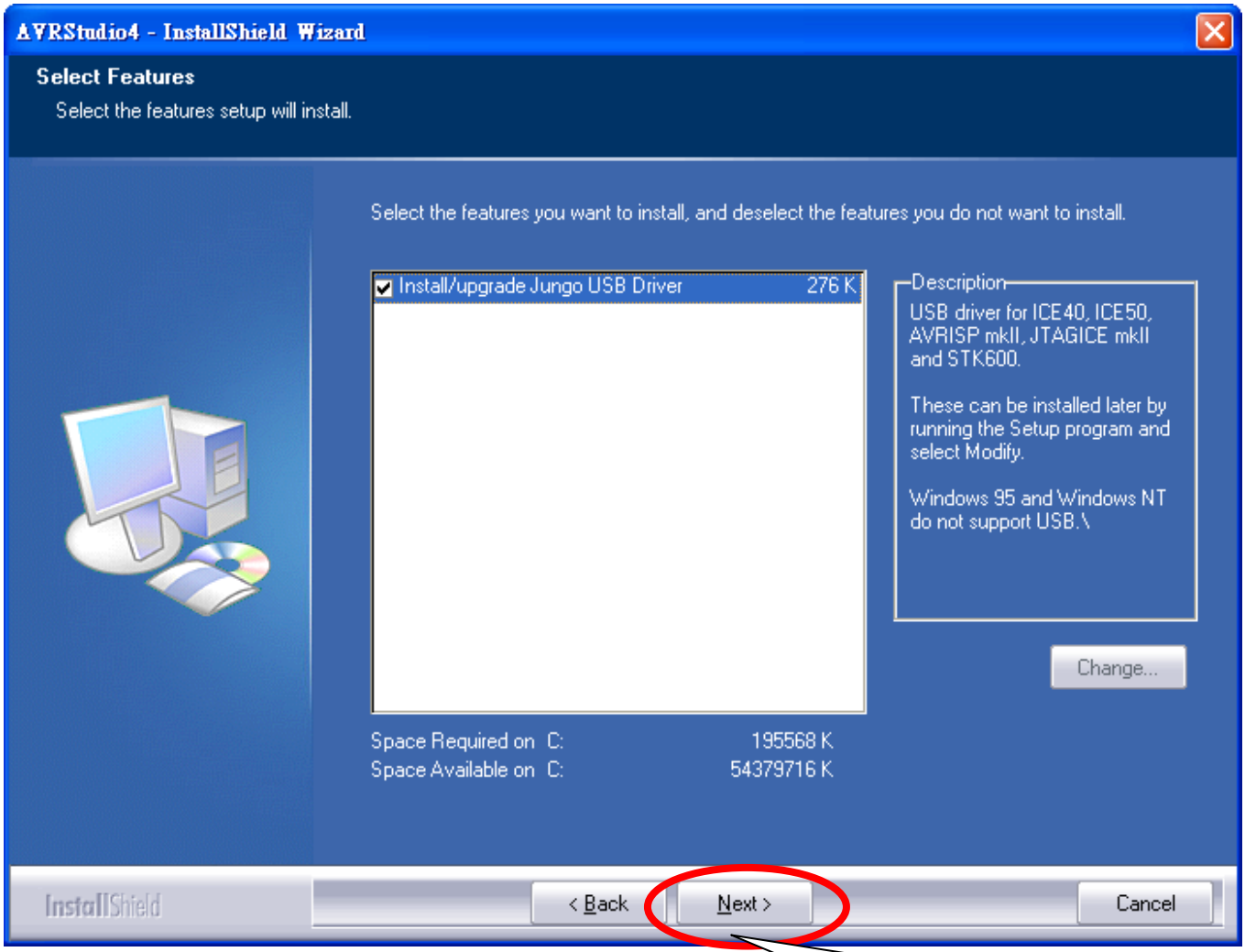
Step2: install AVR studio on designer PC



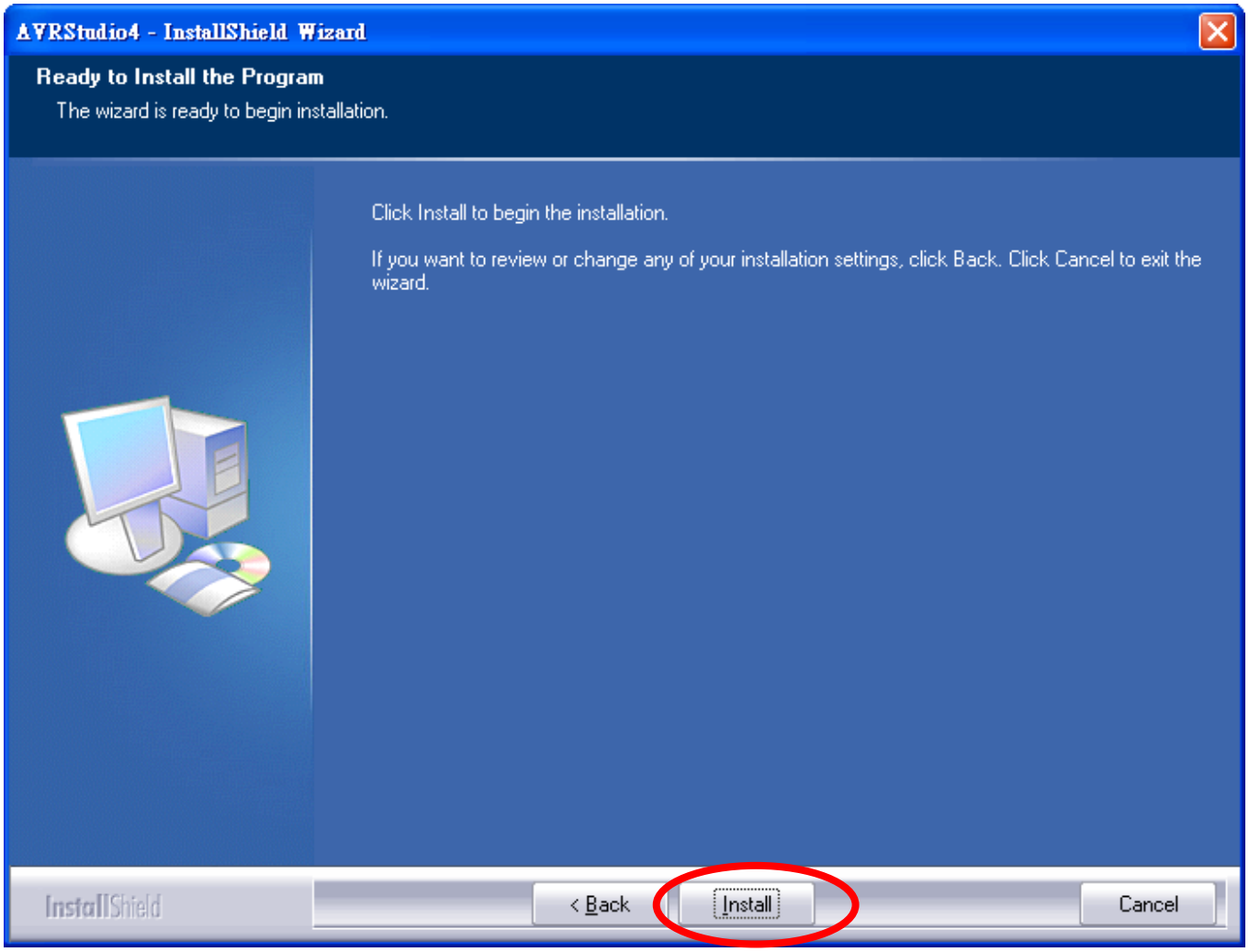

AVRStudio4 - InstallShield Wizard
Welcome to the InstallShield Wizard for AVRStudio4
The InstallShield Wizard will install AVRStudio4 on your computer. To continue, click Next.
InstallShield
< Back
Next >
Cancel


AVRStudio4 - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
Welcome to AVR Studio from Atmel Corporation.
AVR Studio is a Development Tool for the entire AVR family of microcontrollers, including tinyAVR, megaAVR, picoPower, and XMEGA AVR devices. The AVR Studio is free of charge and may be freely copied and distributed in its original form.
AVR Studio enables the user to fully control execution of programs on the AVR In-Circuit Emulator or on the included AVR Instruction Set Simulator. AVR Studio supports source level execution of Assembly and C/C++ programs assembled with the Atmel Corporation's included AVR Assembler or tools from 3rd party vendors.
AVR Studio runs under Microsoft Windows 98, Windows NT, Microsoft Windows 2000, Windows XP and Windows Vista.
AVR Studio is continously developing. In order to get latest upgrades of AVR Studio, please visit our web site
www.atmel.com
and check out the AVR page.
I accept the terms of the license agreement
I do not accept the terms of the license agreement
Print
InstallShield
< Back
Next >
Cancel
Accept terms to continue

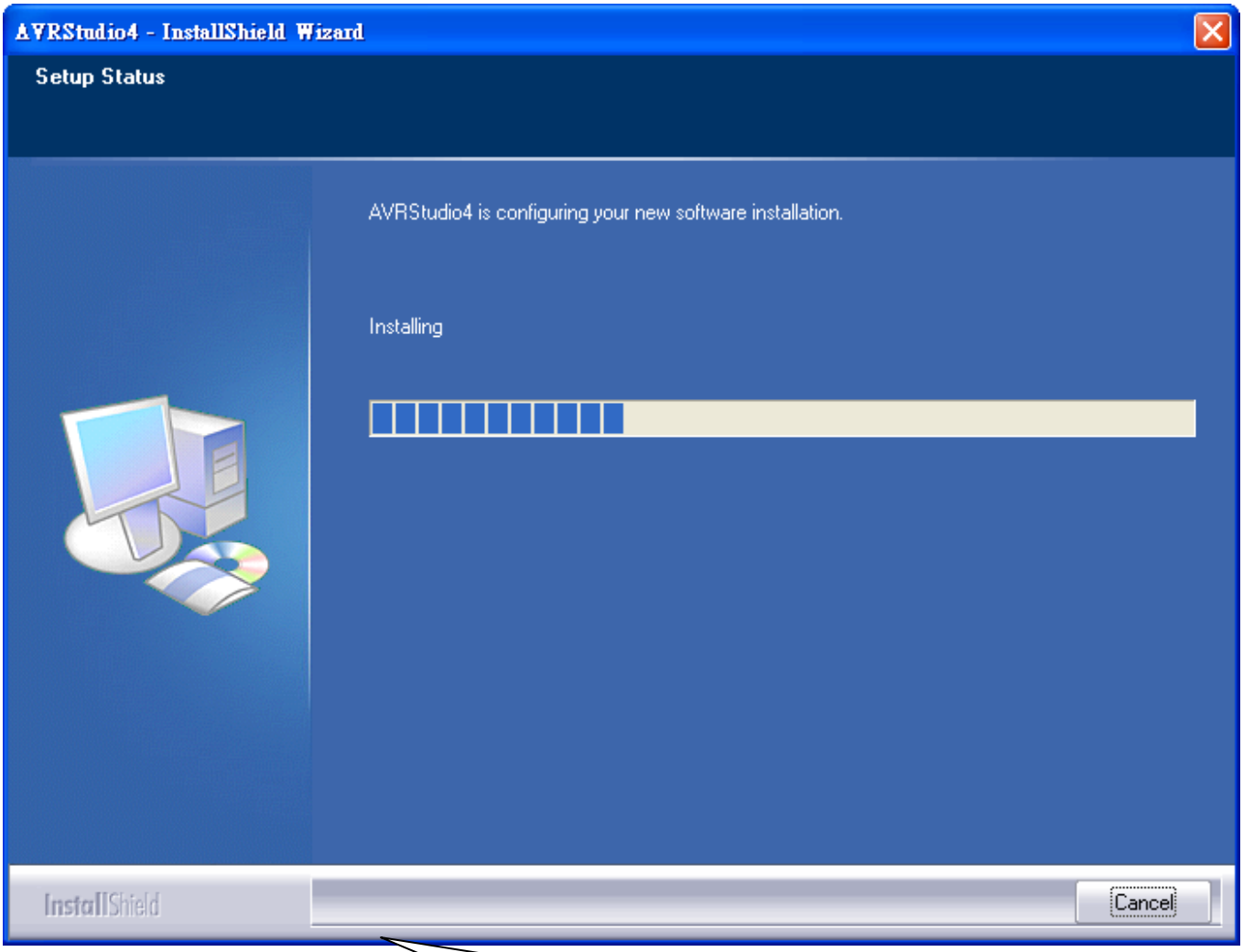

AVRStudio4 - InstallShield Wizard
Choose Destination Location
Select folder where setup will install files.
Install AVRStudio4 to:
C:\Program Files\Atmel\AVR Tools
Change...
InstallShield
< Back
Next >
Cancel
Select folder to install and continue



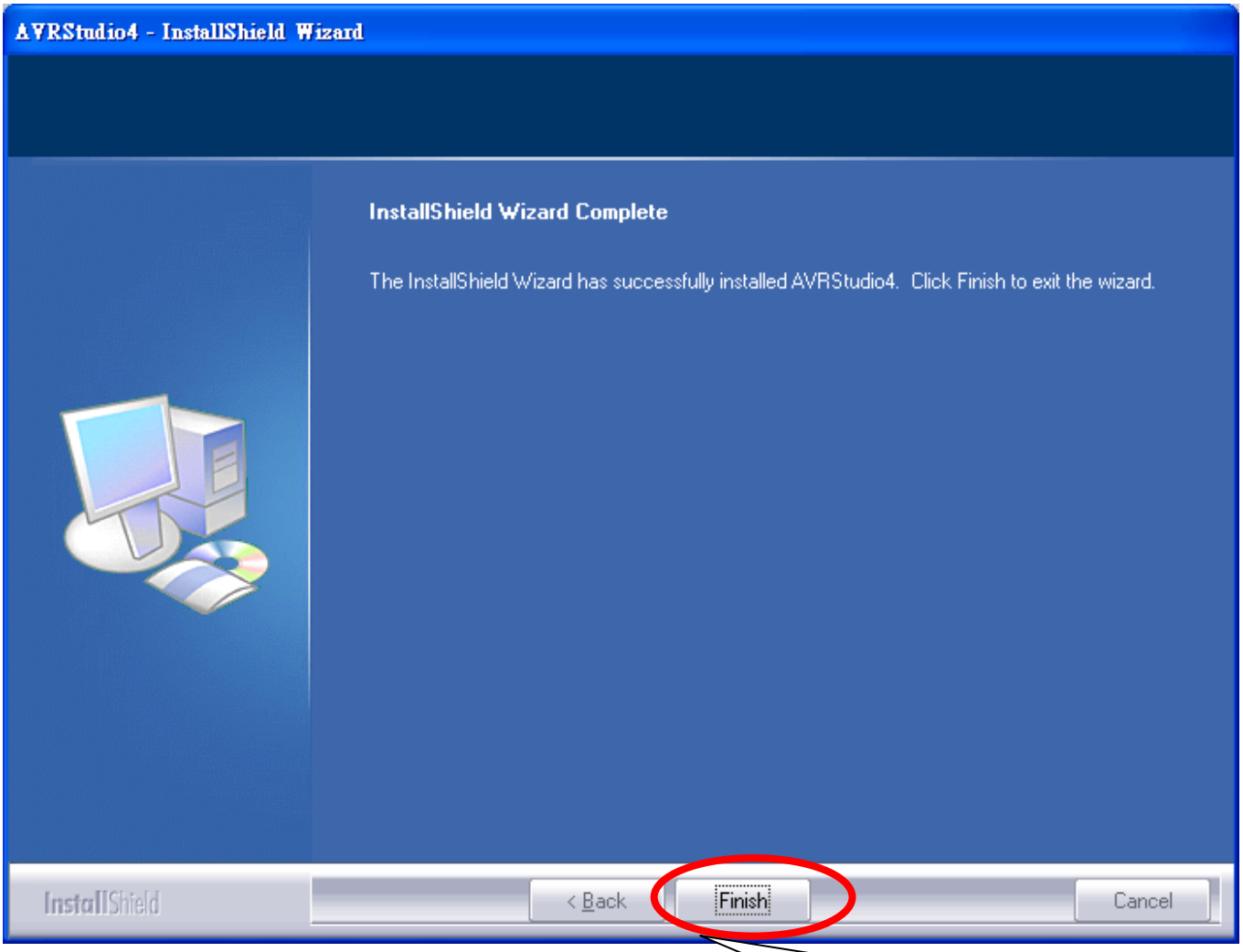
Select features to install and continue



Click **install** to begin installation



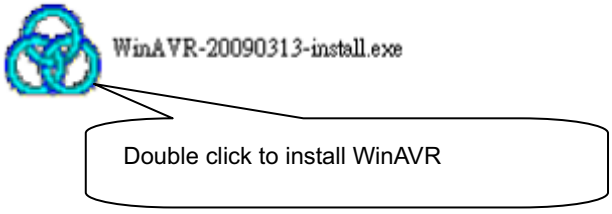
Wait for seconds for installation

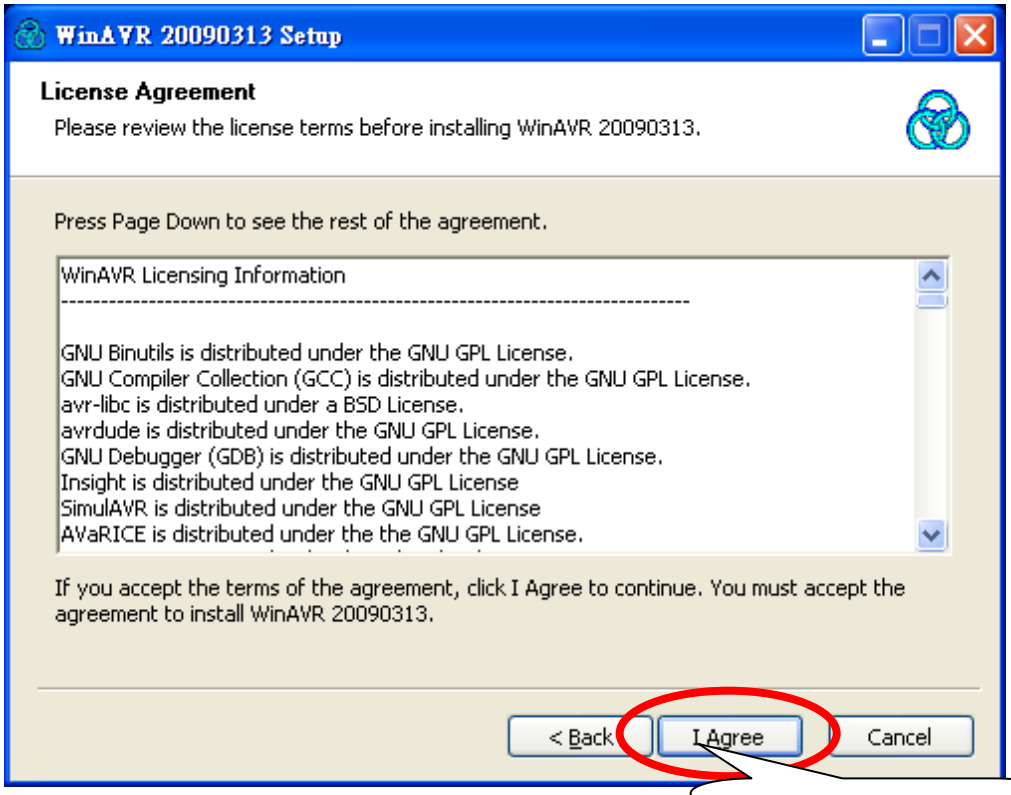


Click **finish** to exit the installation

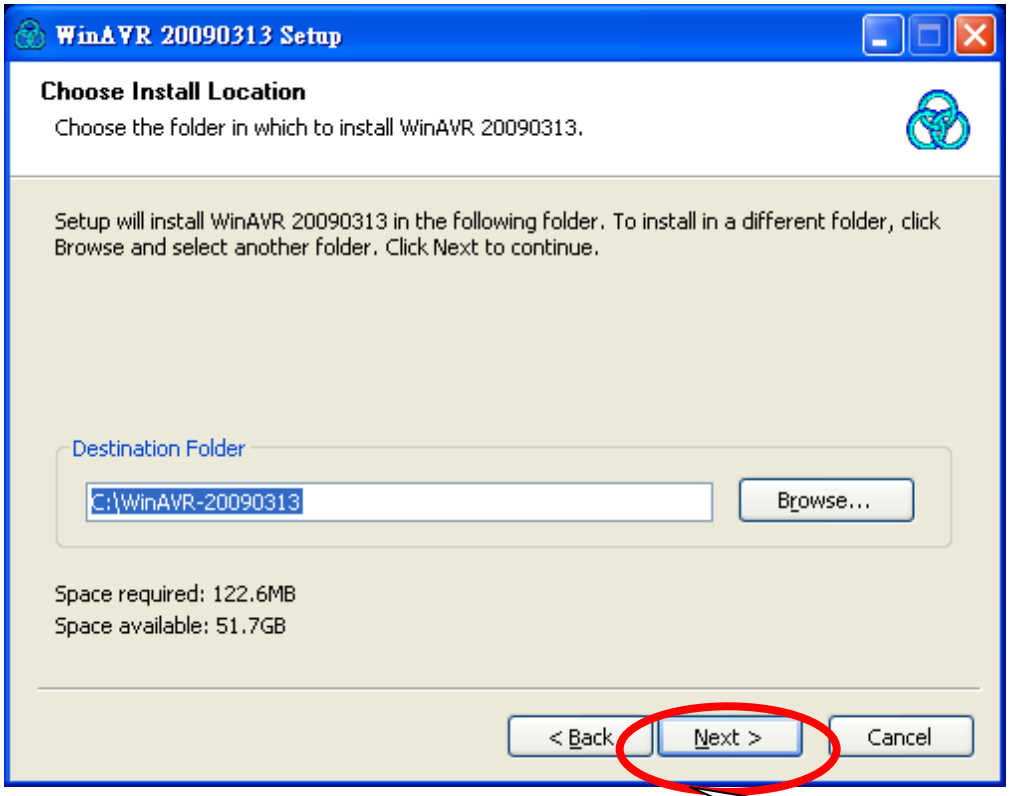
4-1-2 Additional tool for C language

For C language designers, additionally please download and install AVR gcc from
http://sourceforge.net/project/downloading.php?group_id=68108&filename=WinAVR-20090313-install.exe&a=6759369

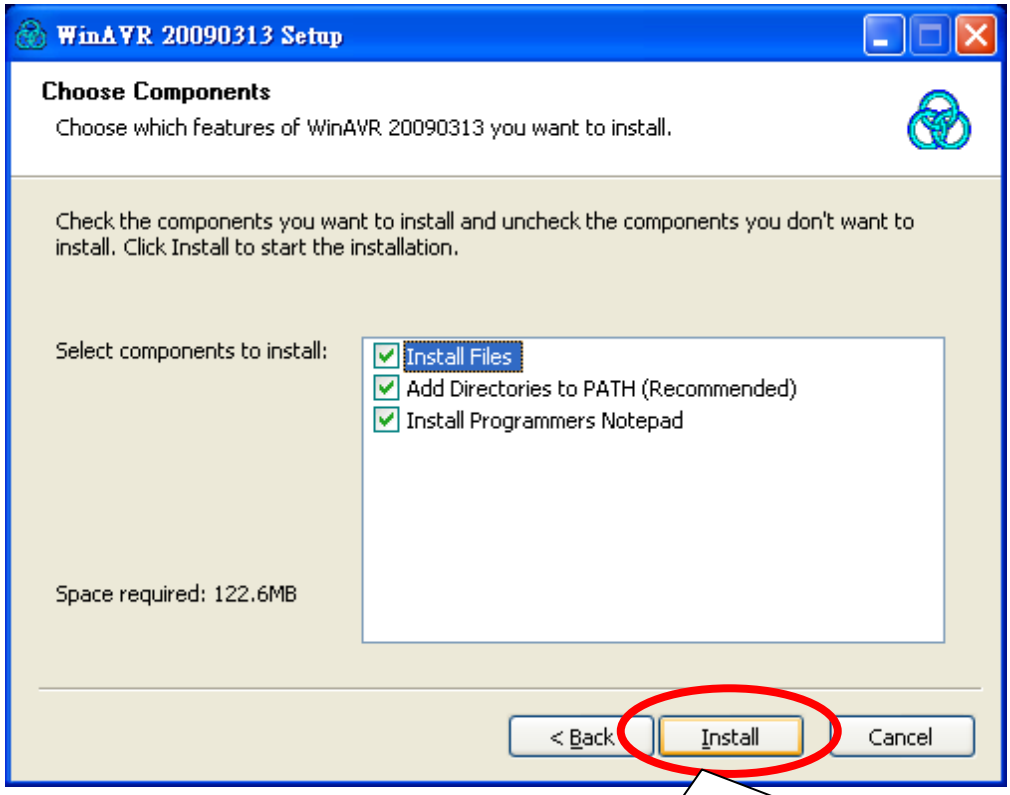




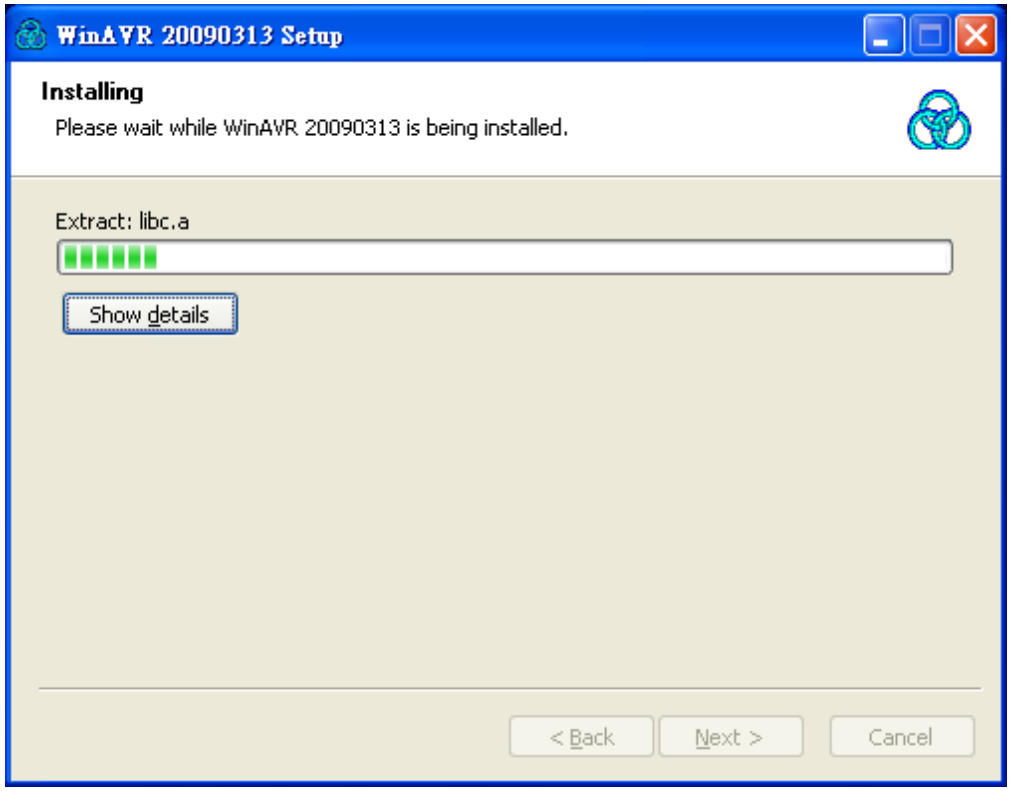
Click **I agree** to continue



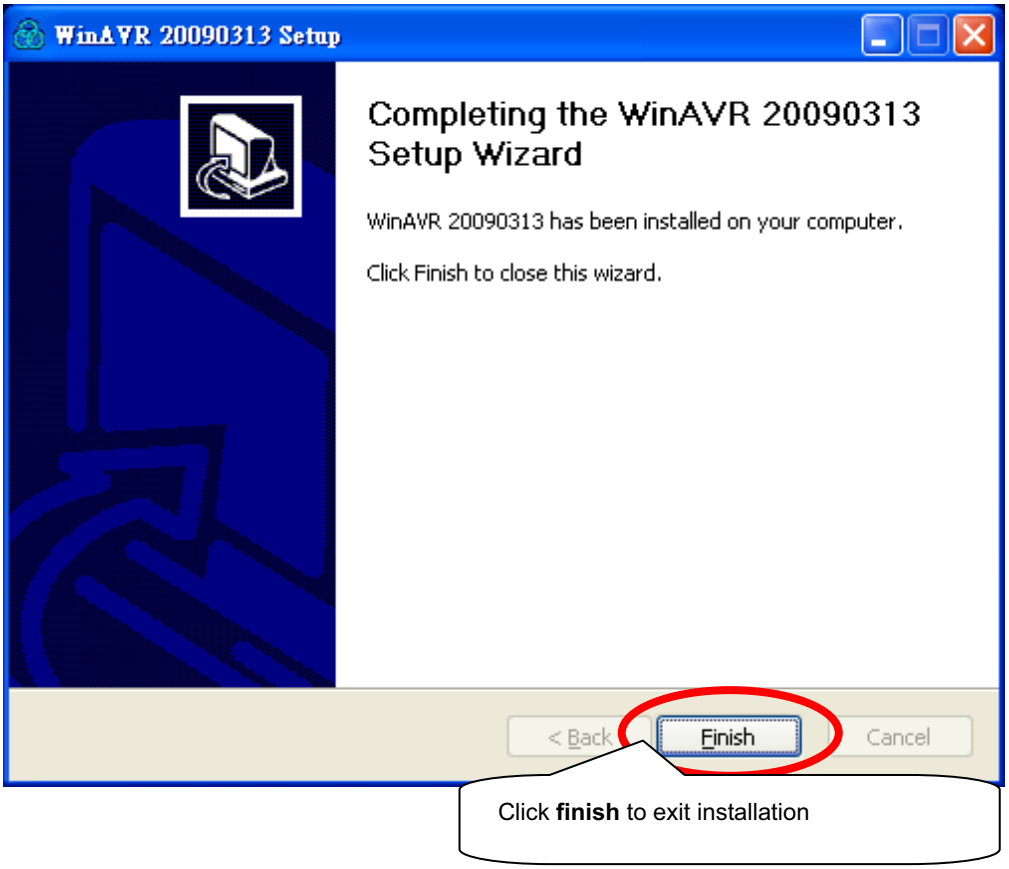
Choose folder to continue



Choose components to install



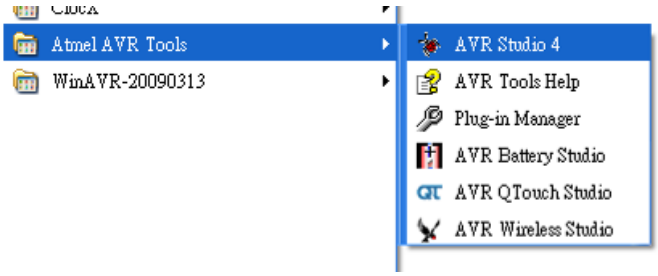
Wait for seconds for installation

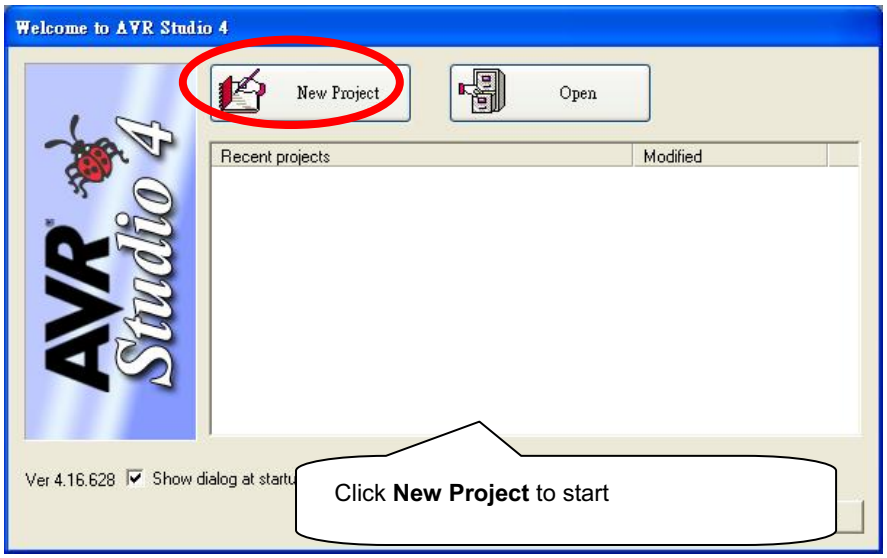


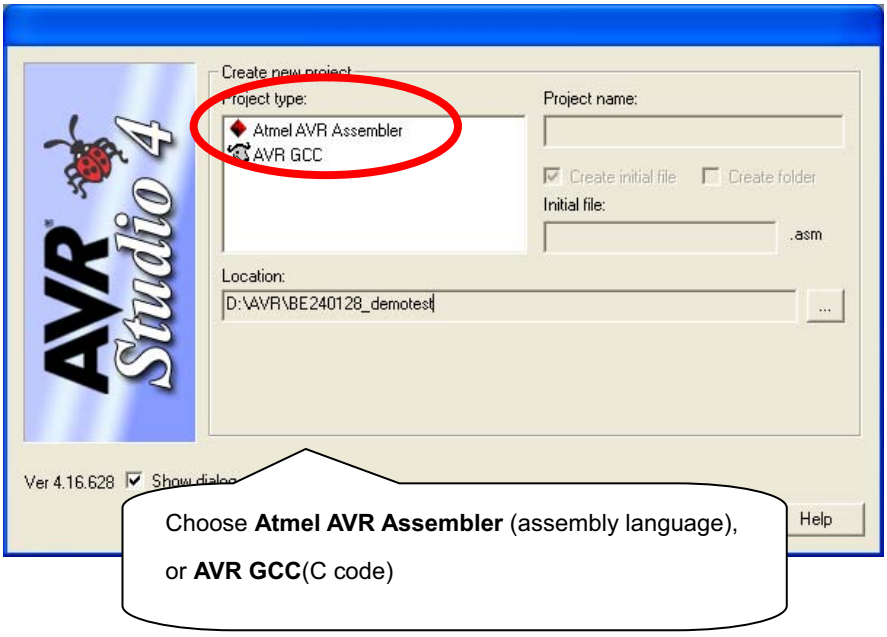
**** AVR Studio and AVR gcc software are also available on Bolymin utility disk or Bolymin website.**

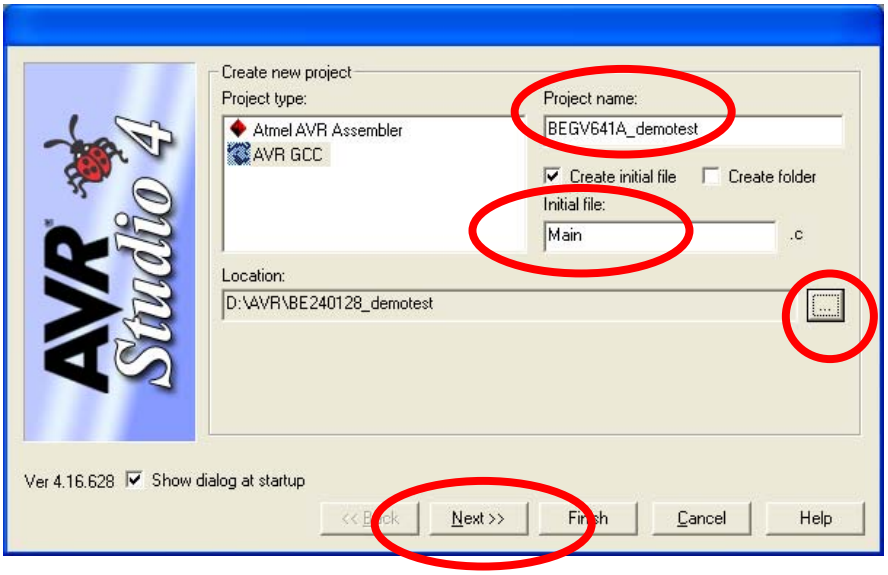
4-2 Execute AVR Studio 4.16 on designer PC

Start→All programs→Atmel AVR Tools→AVR Studio 4

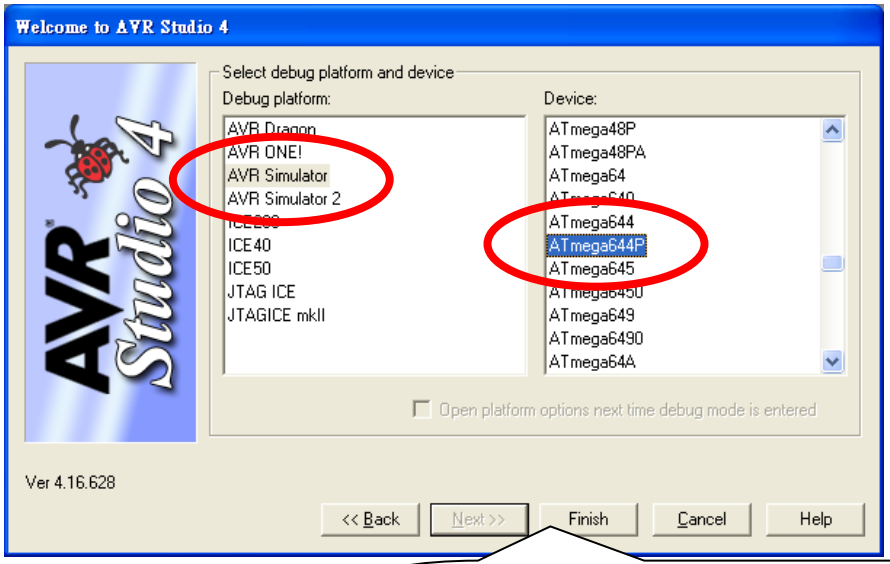




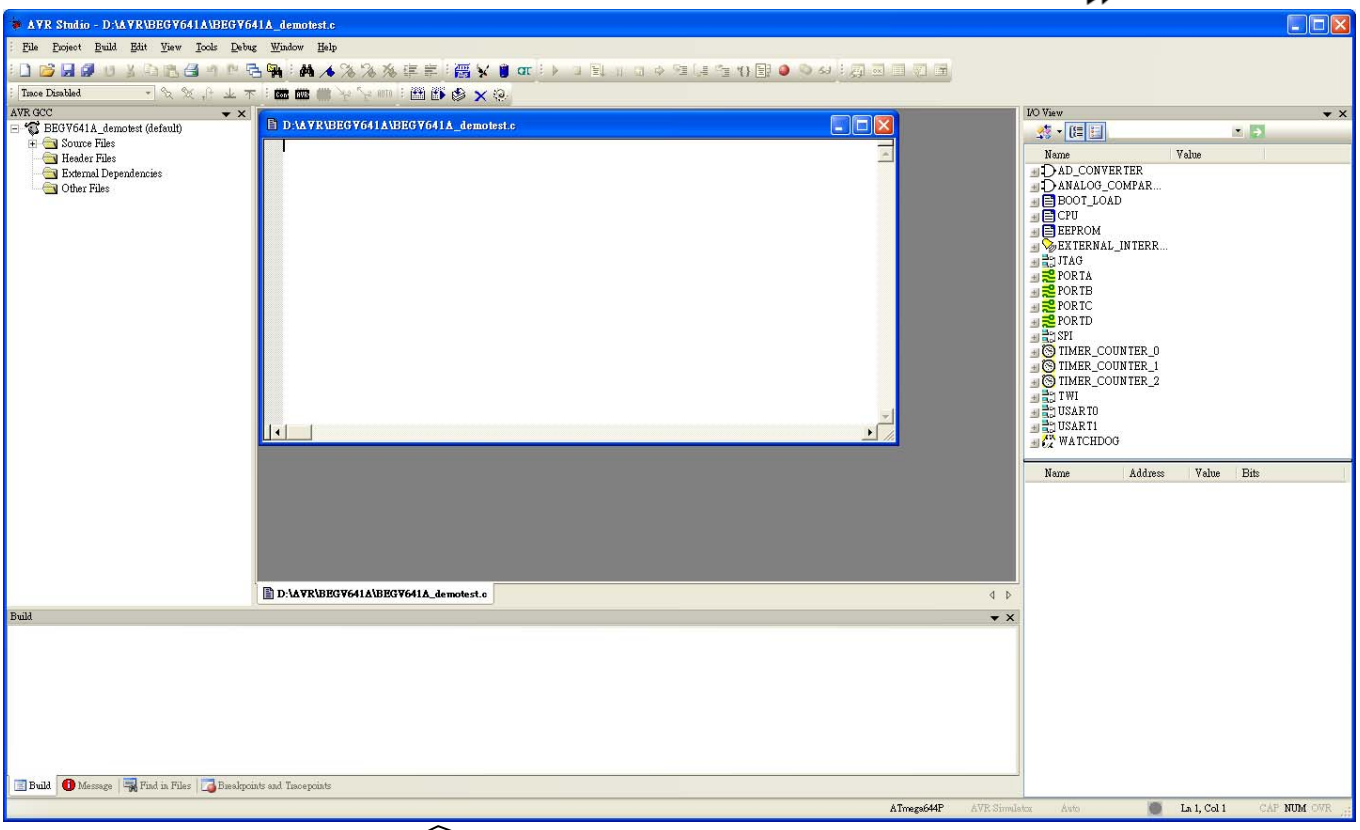




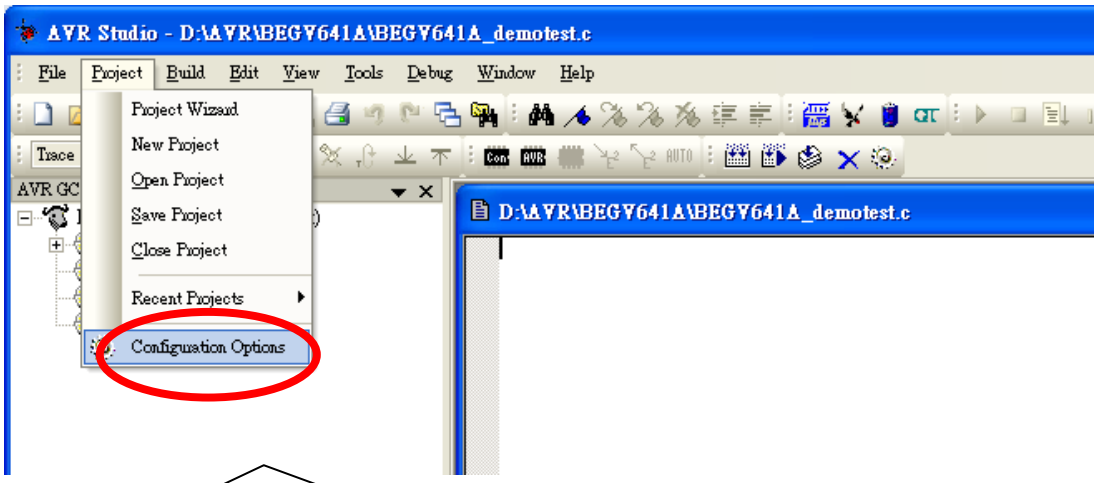
Choose **AVR GCC** here (All Bolymin utility drivers are designed with C code), and key in Project Name and Initial file.



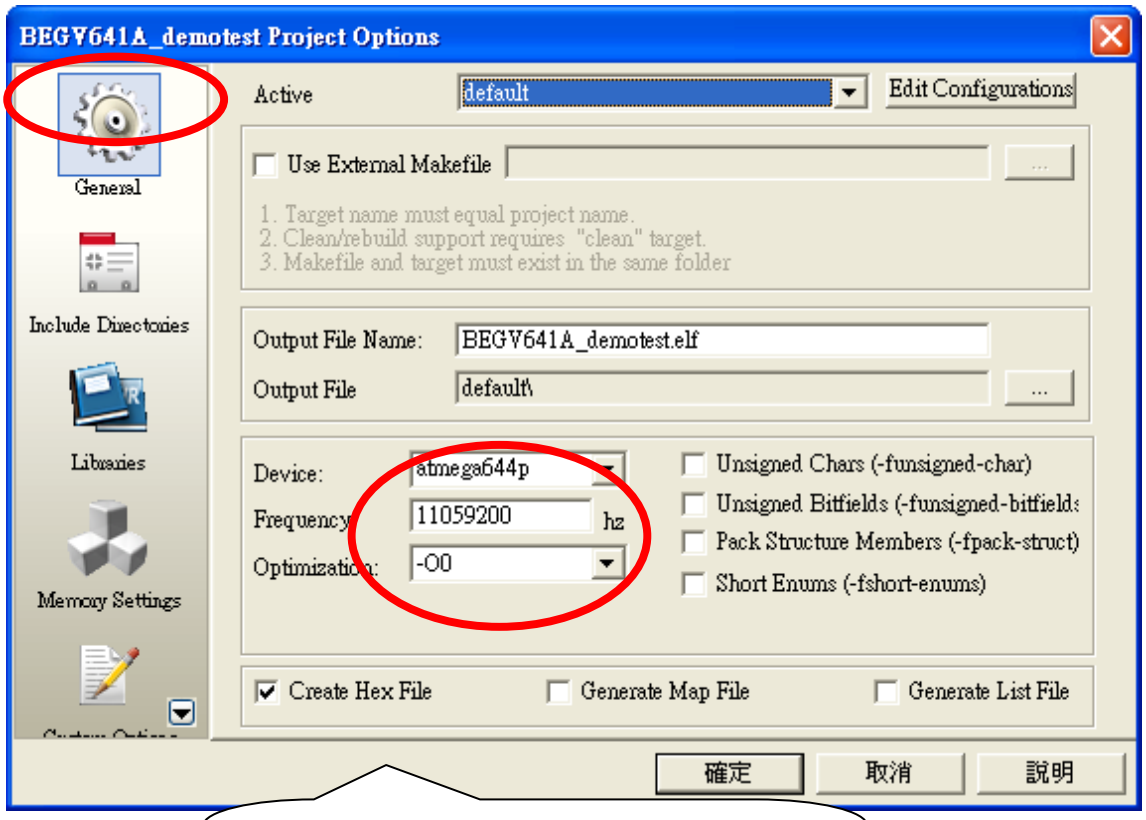
Choose **AVR Simulator**, **ATmega644P**, and click **Finish** to continue



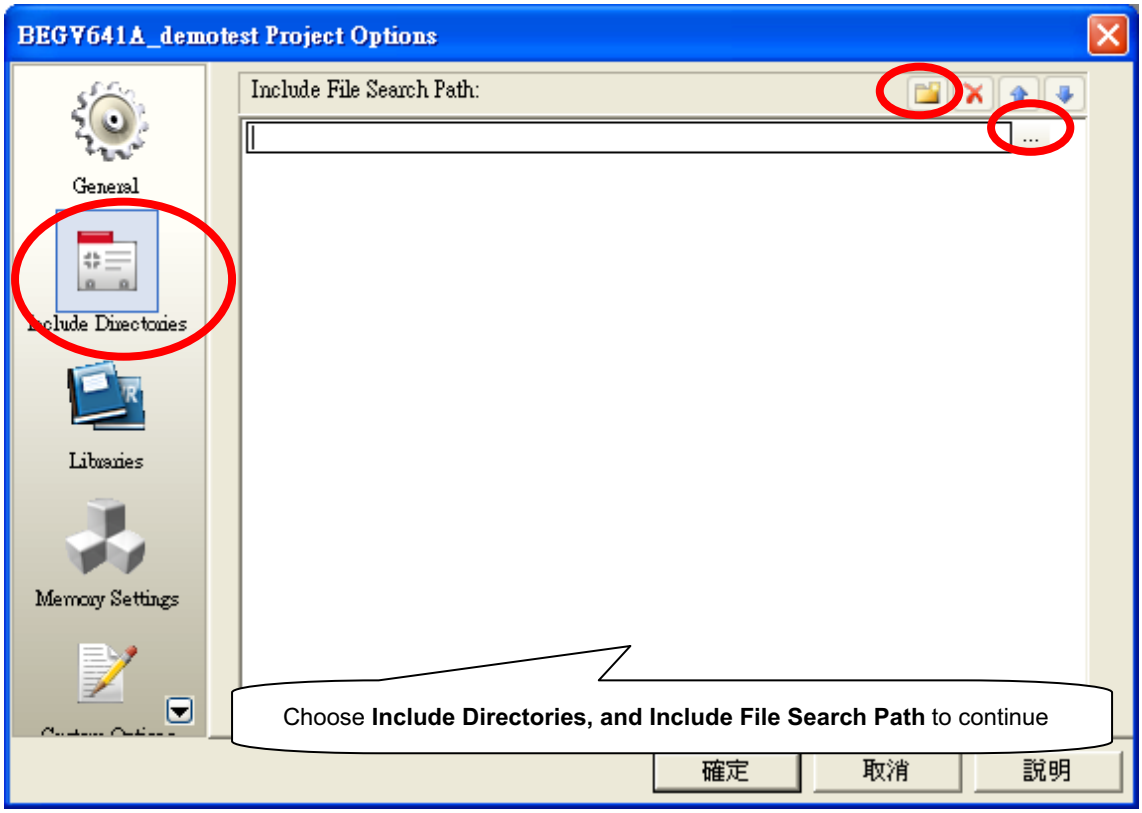
Here you can see software design screen, yet no hurry to start software design yet.



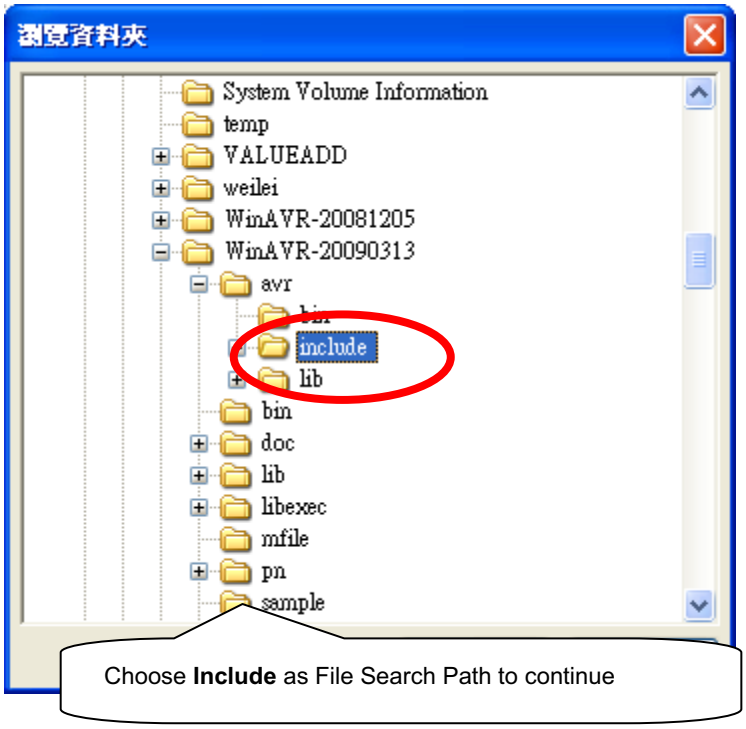
Choose **Project →Configuration Options** to continue

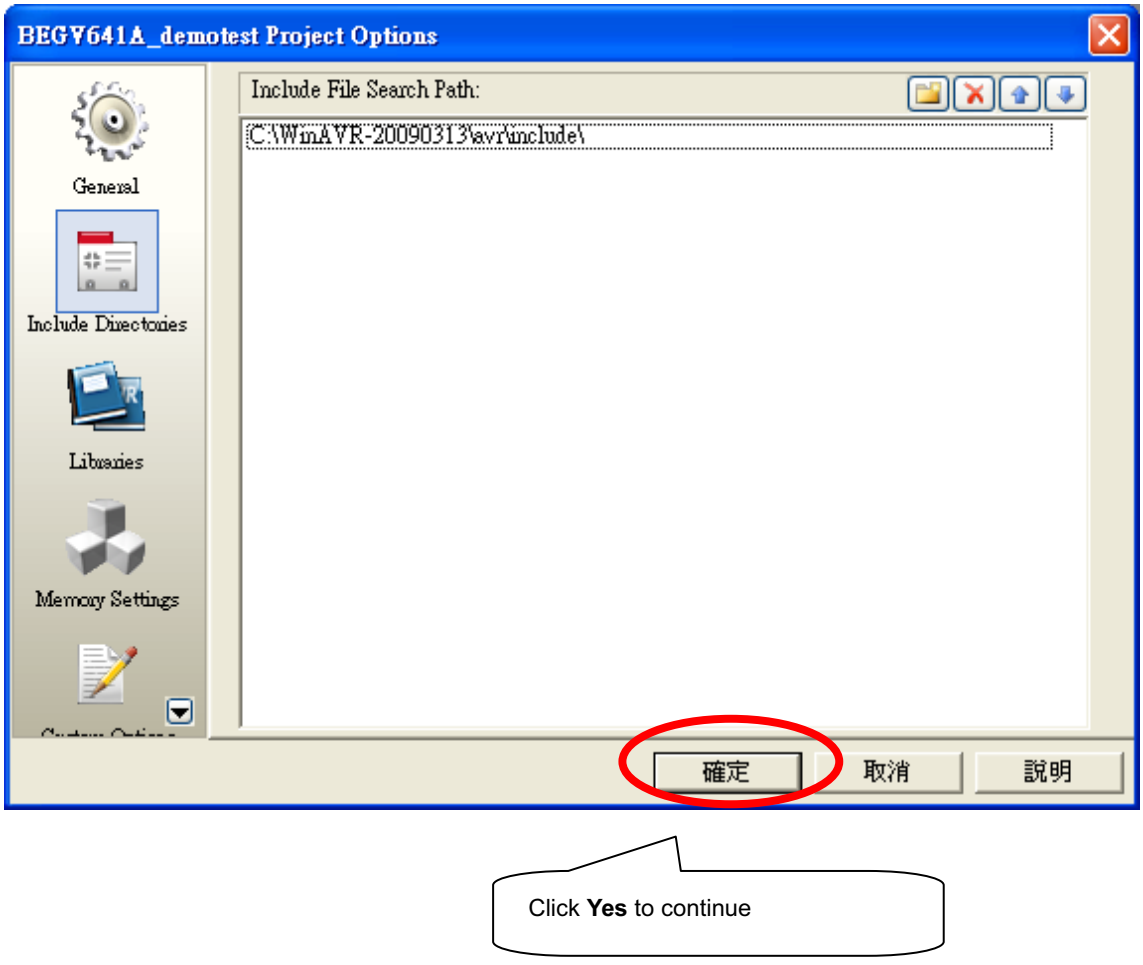


Choose **General** and key in necessary information to continue

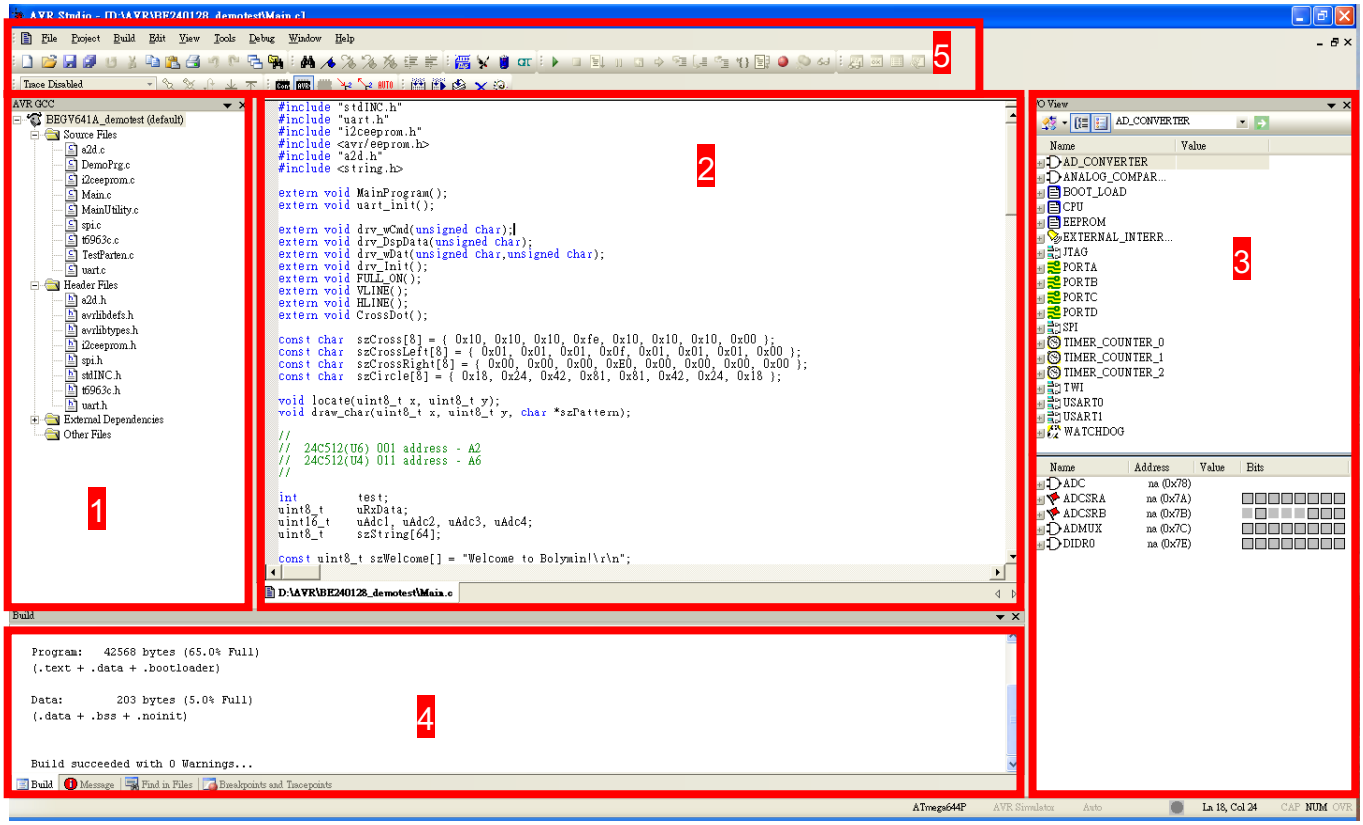


Choose **Include Directories, and Include File Search Path** to continue

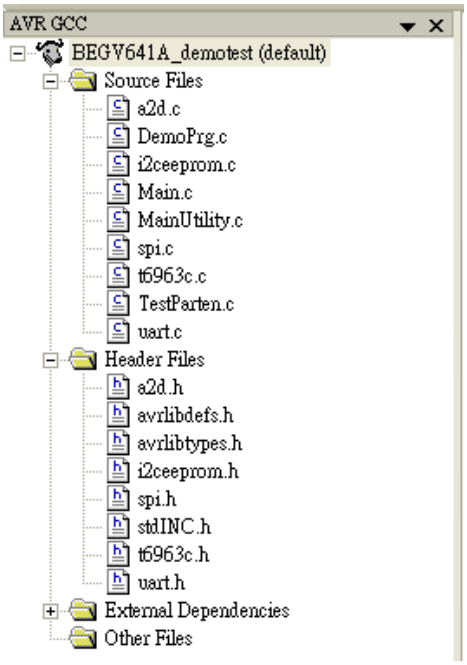

瀏覽資料夾
System Volume Information
temp
VALUEADD
weilei
WinAVR-20081205
WinAVR-20090313
avr
include
lib
bin
doc
lib
libexec
mfile
pn
sample
Choose **Include** as File Search Path to continue


BEGV641A_demotest Project Options
General
Include Directories
Libraries
Memory Settings
Include File Search Path:
C:\WinAVR-20090313\avr\include\
確定
取消
說明
Click **Yes** to continue

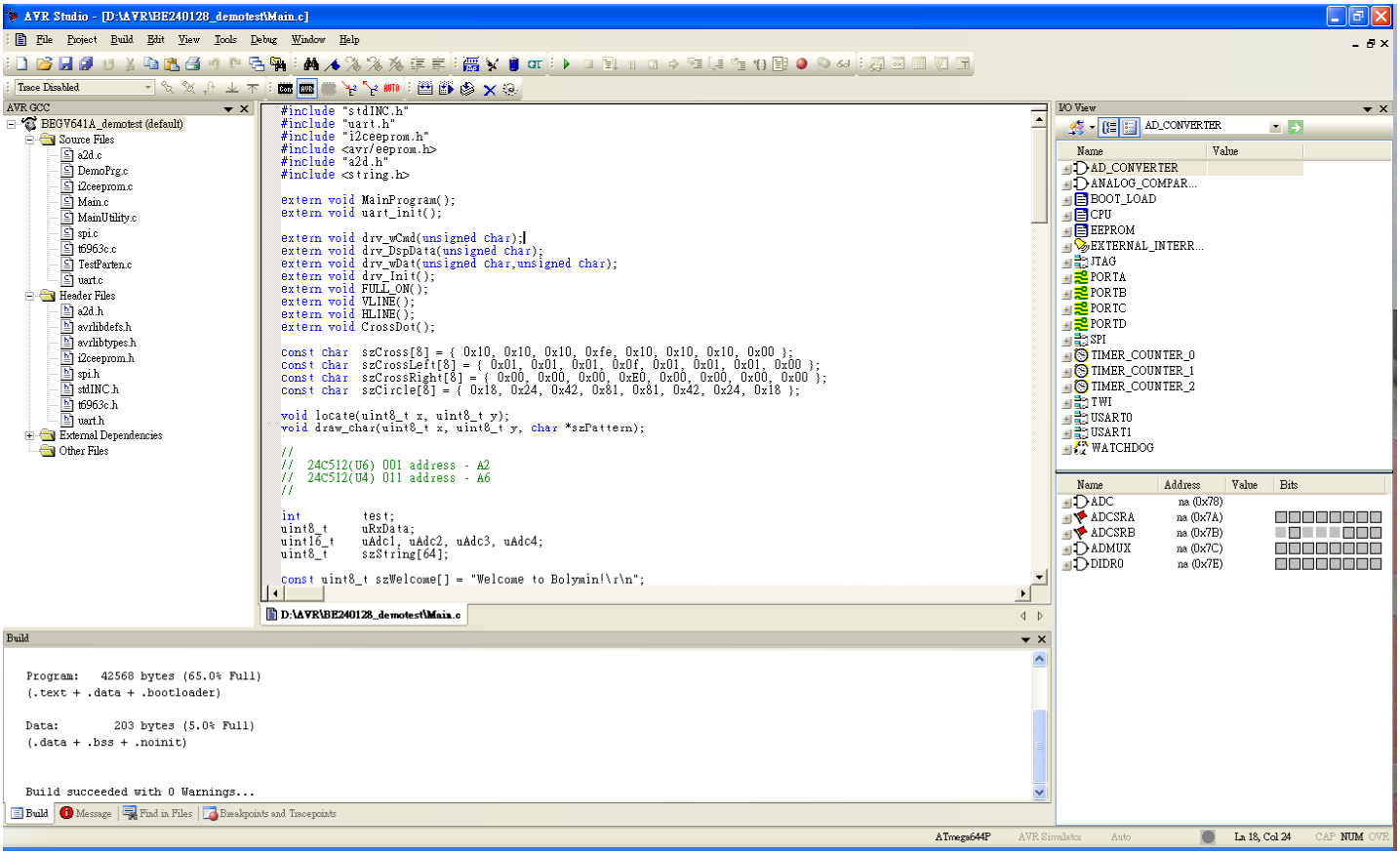
After executing AVR Studio 4.16 on PC, designer can see key information on 5 windows:



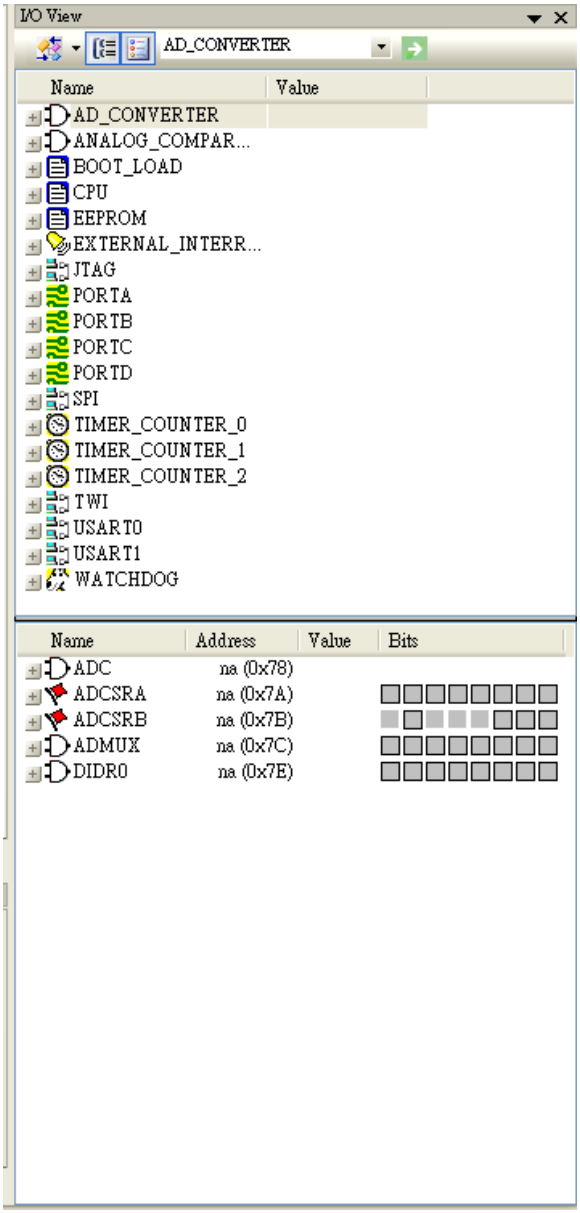
[1]. Project management window



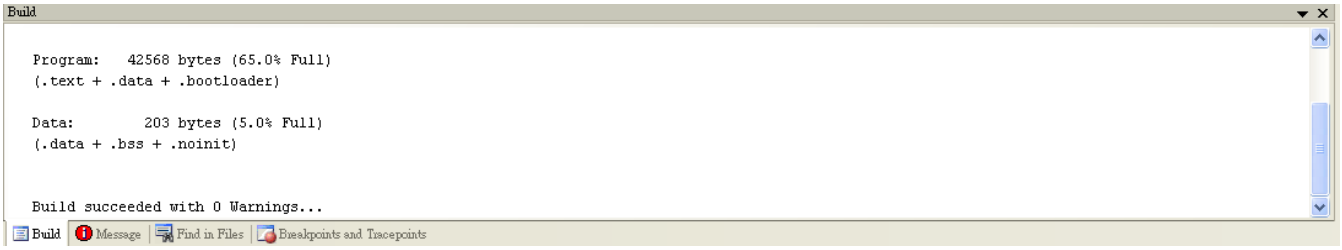
[2]. Program editing window



[3]. I/O view window



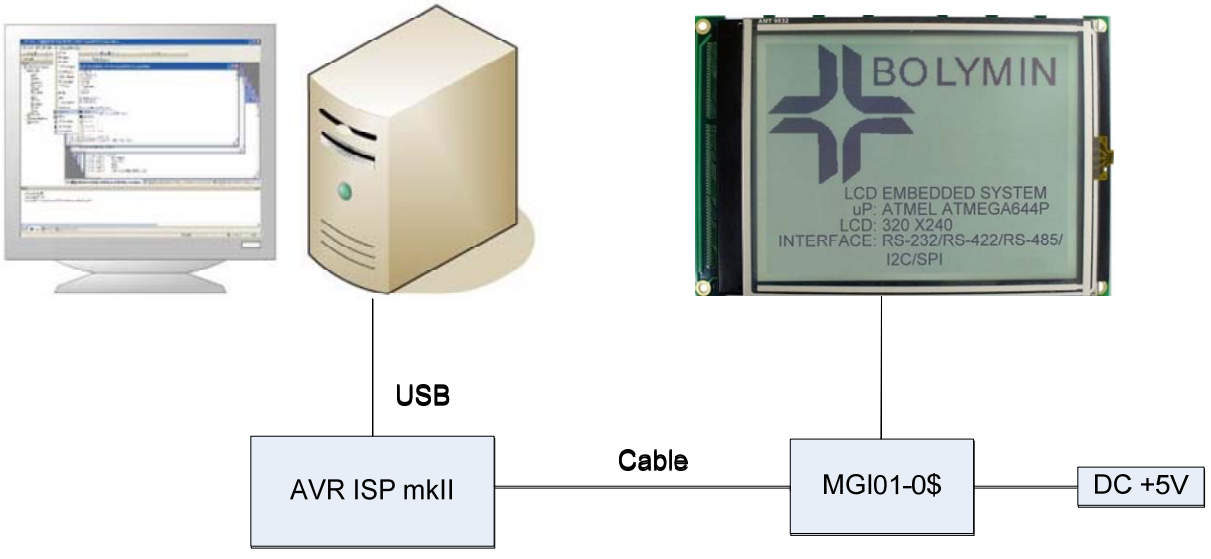
[4]. Message window



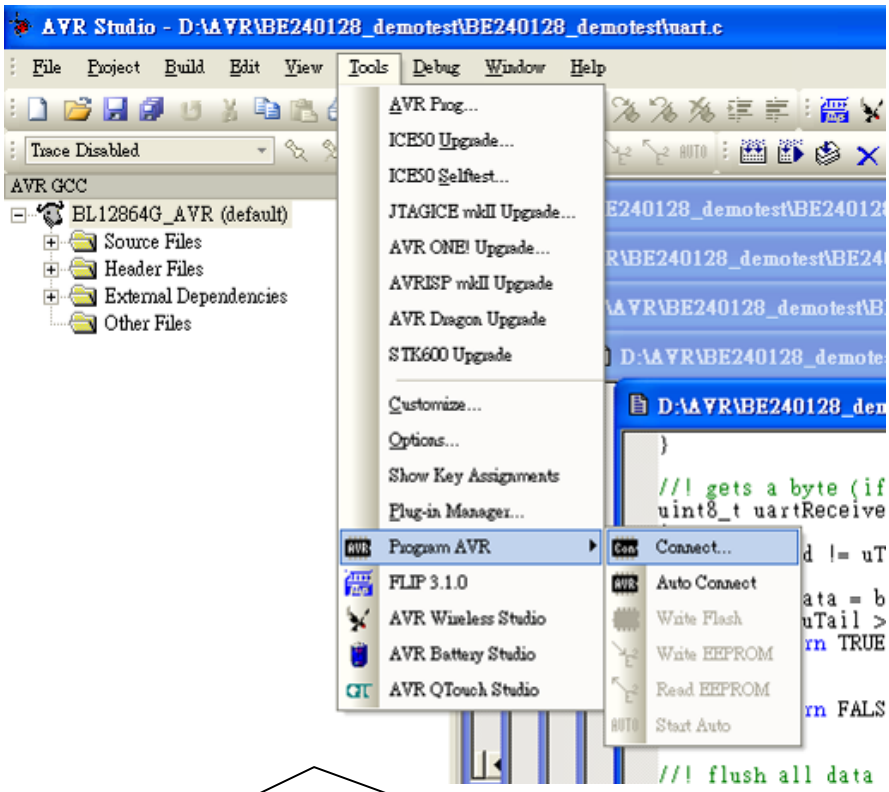
[5]. Toolbars

4-3 In-System Programmer AVR ISP mkII

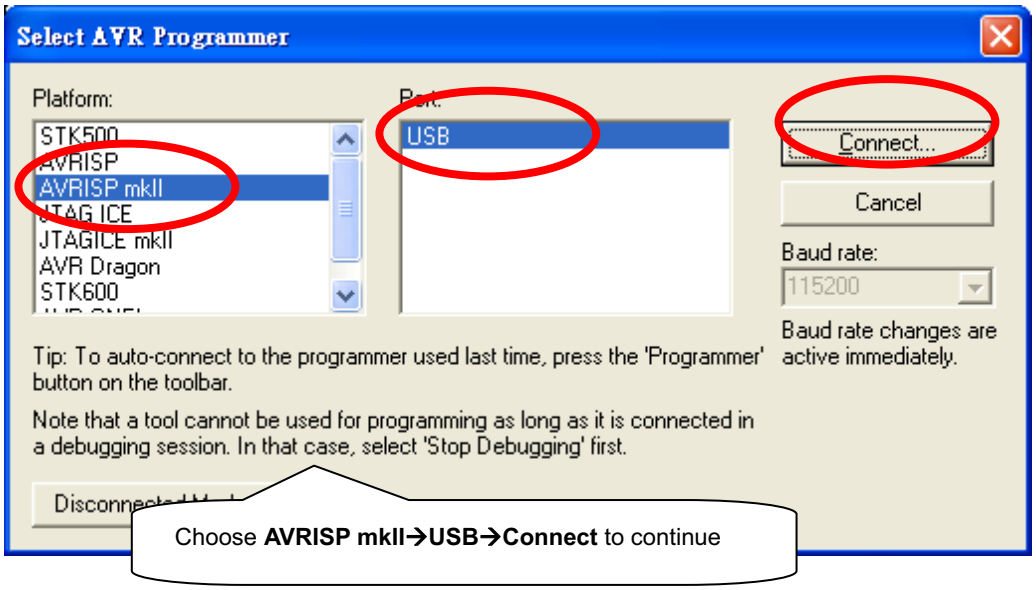
** Inside mkII box(1), designer will get DVD(2), mkII device(3), and USB cable(4).
Please be sure that all parts are packed inside.



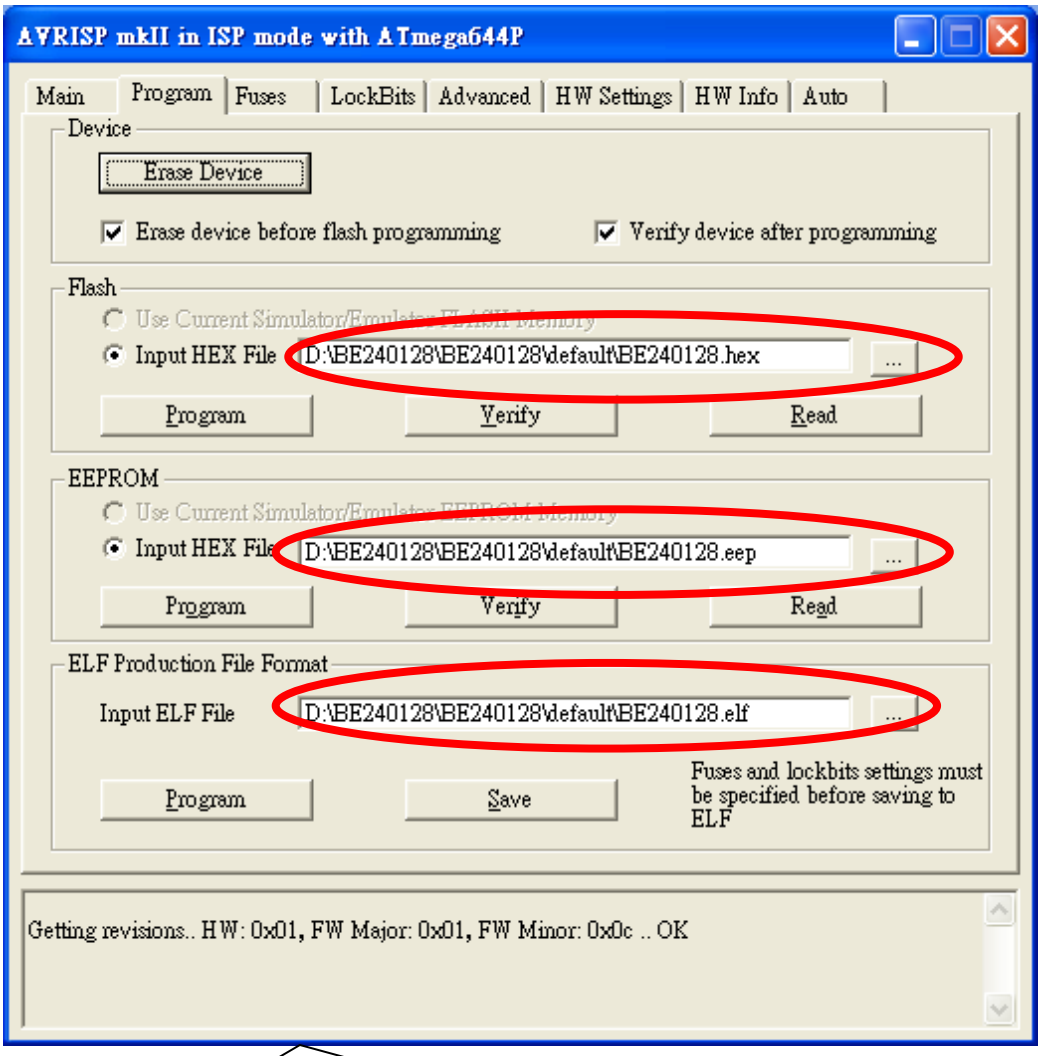
Steps:
1.Get mkII connected to PC



Choose **Tools→Program AVR→Connect** to continue



Choose **AVRISP mkII→USB→Connect** to continue

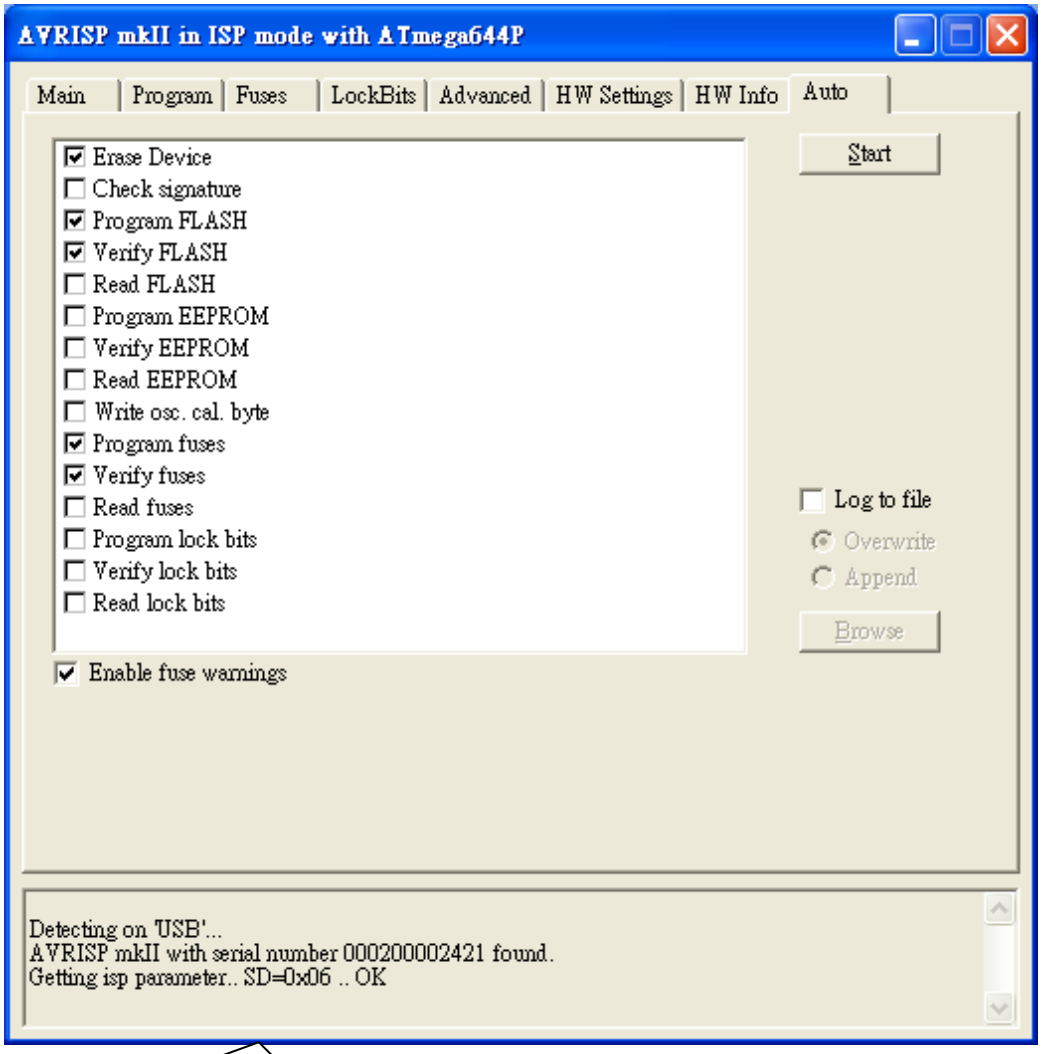


Choose Hex file for ATmega644p flash → **Program**

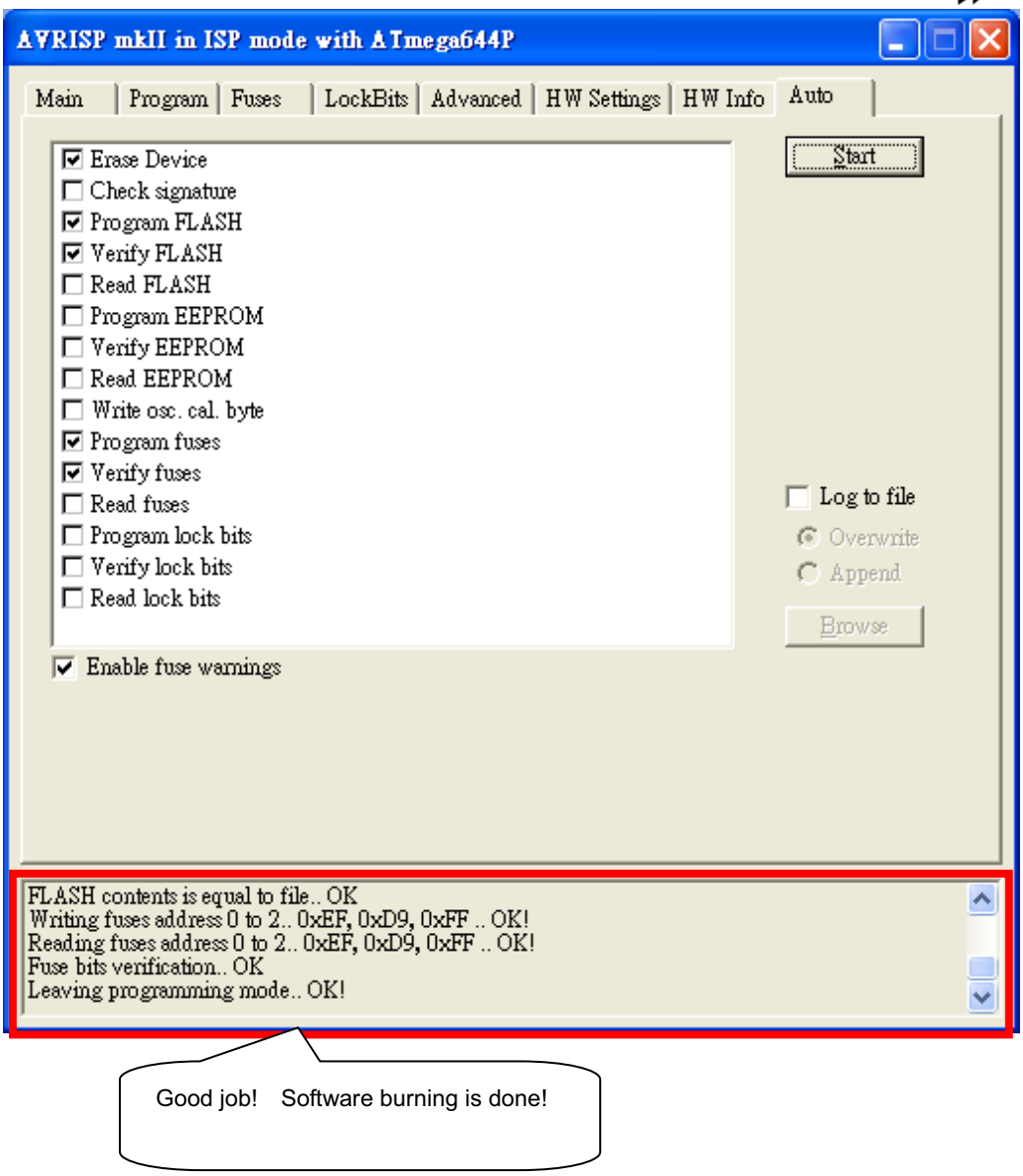
Choose Hex file for ATmega644p EEPROM→**Program**

Choose ELF file for fuses and lockbits→ **Program**

Start software burning

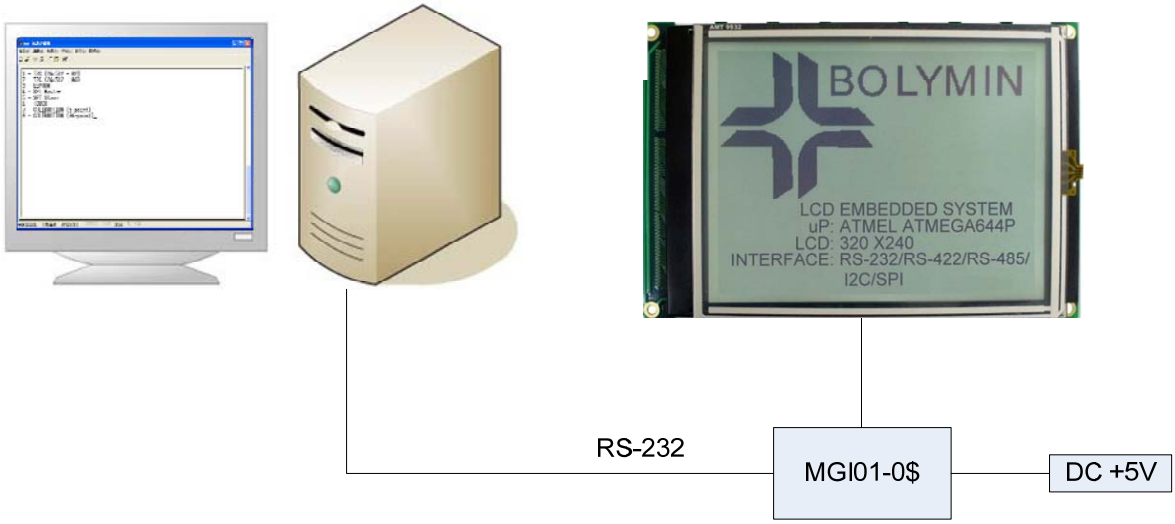


Choose **Auto** and necessary configure and click **Start** to program

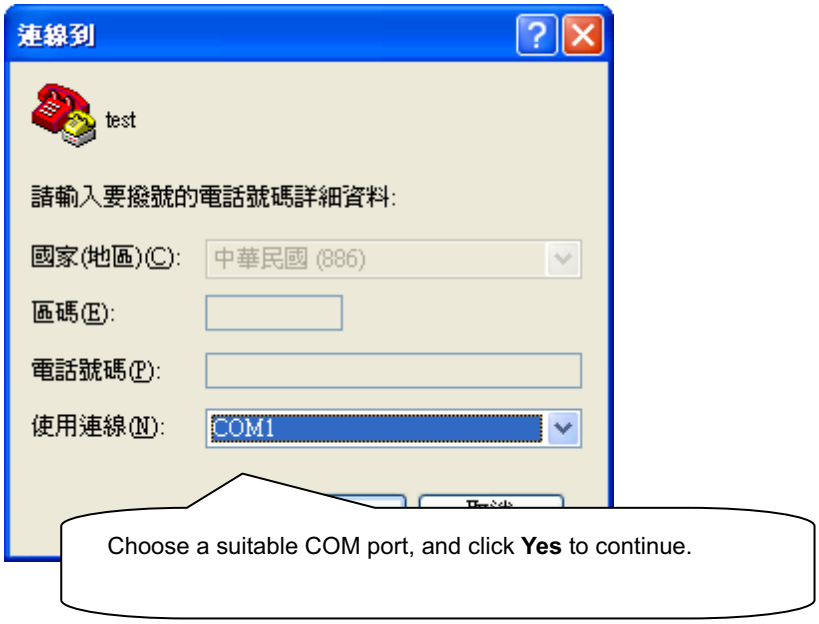


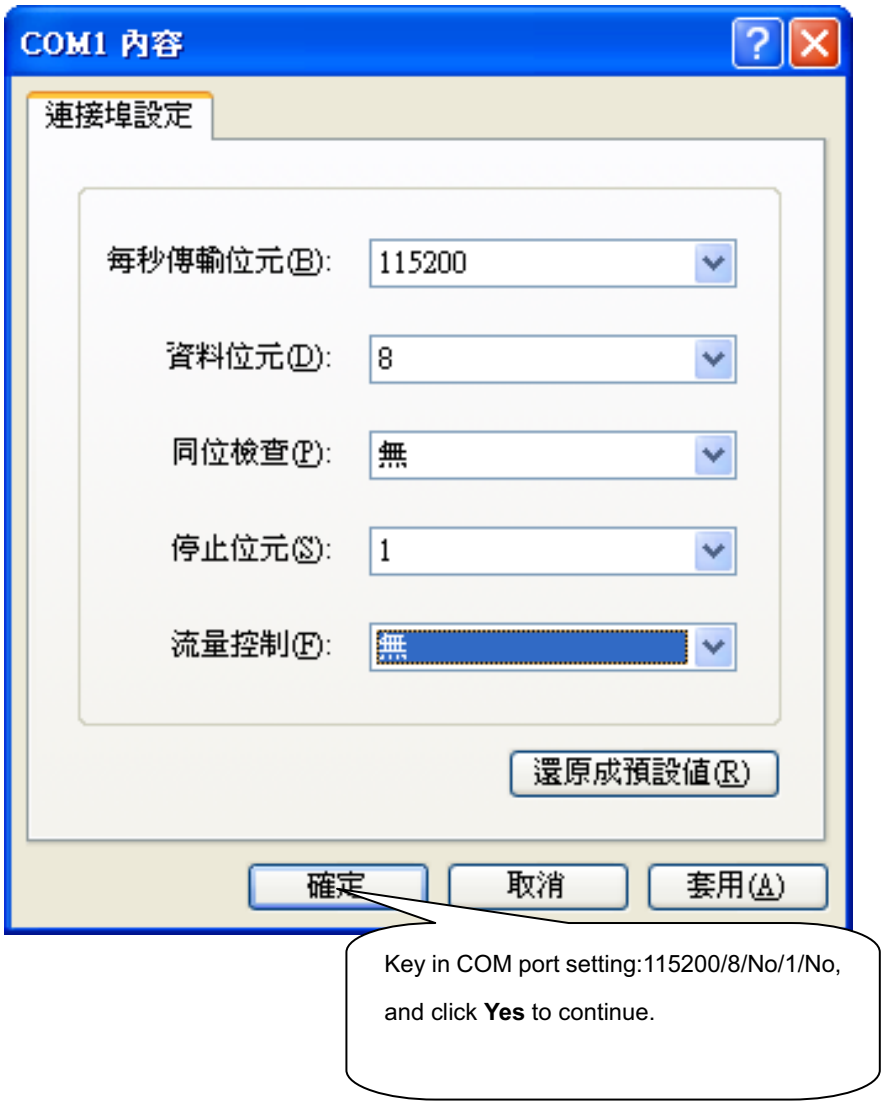
Good job! Software burning is done!

Product Function Verify

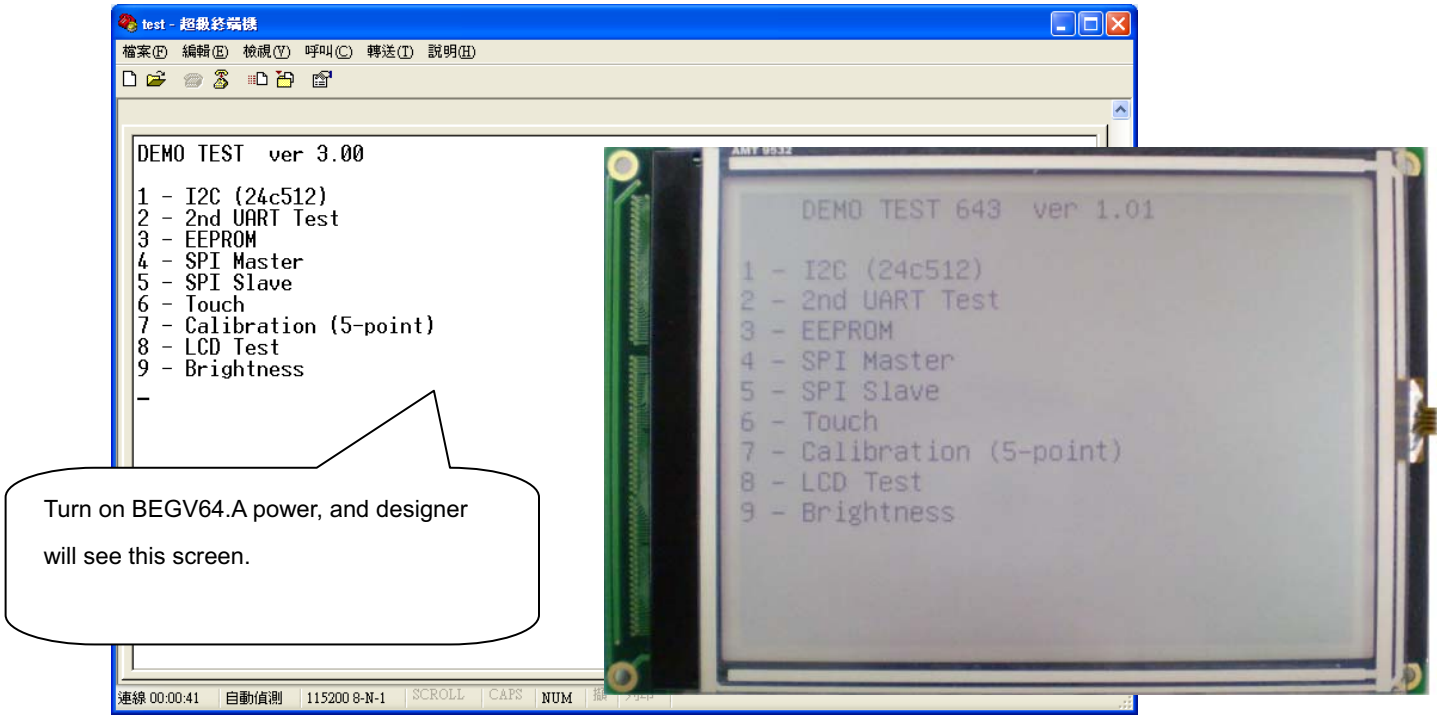


On PC: Start→All programs→Telecommunication→Hyper Terminal





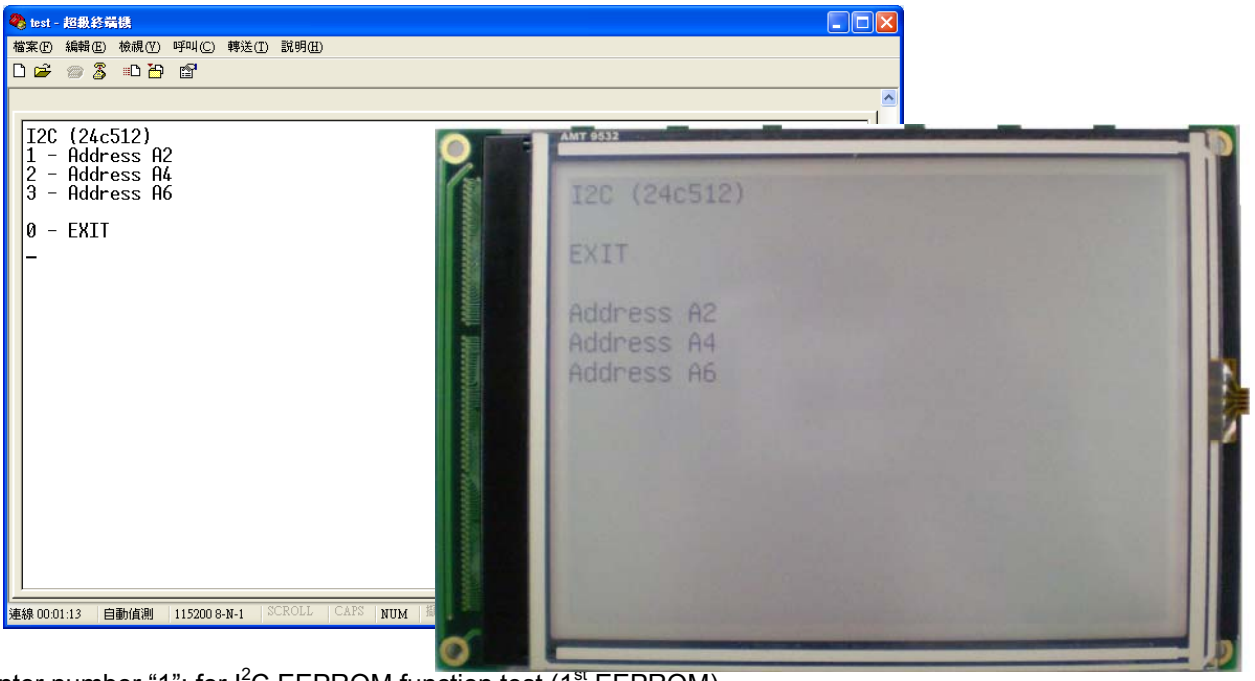
Key in COM port setting:115200/8/No/1/No, and click **Yes** to continue.



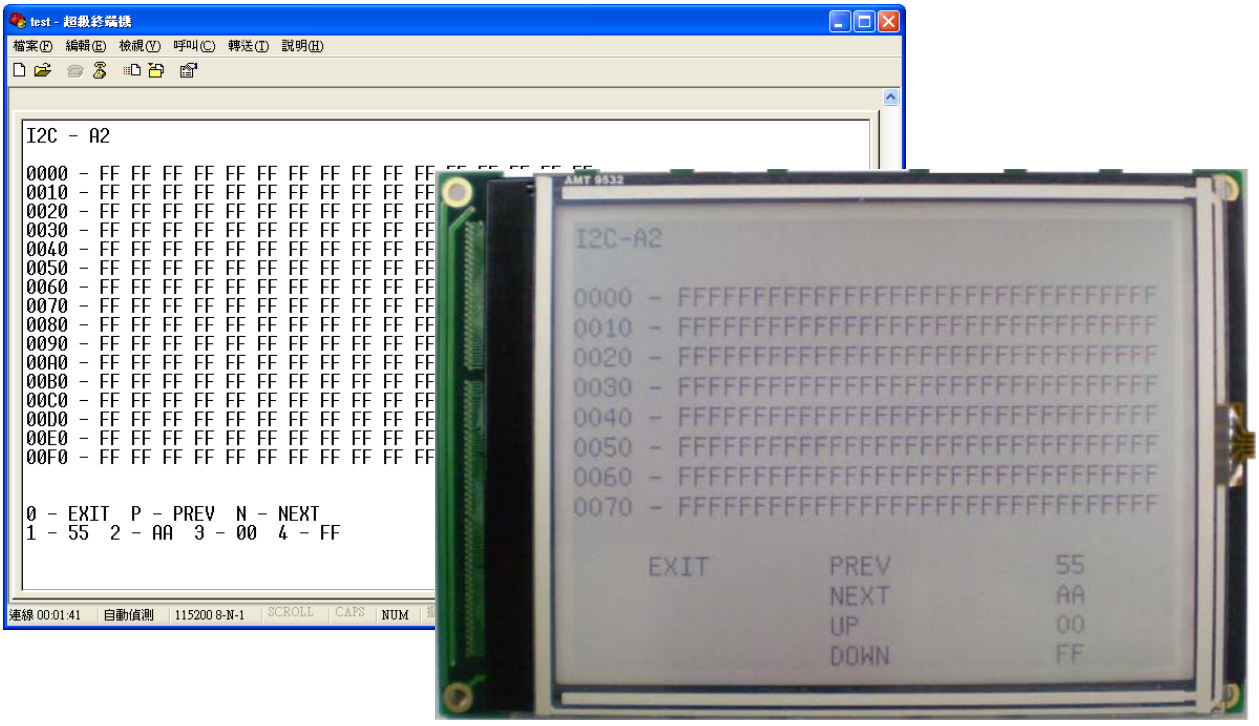
After seeing above screen, designer can operate on PC & BEGV643A:

A). Enter number “1”: for I^{2}C EEPROM function test (1st 2nd 3rd EEPROM)
B). Enter number “2”: for 2nd UART function test
C). Enter number “3”: for ATmega644P internal EEPROM function test
D). Enter number “4”: for 4-wire SPI Master function test
E). Enter number “5”: for 4-wire SPI Slave function test
F). Enter number “6”: to touch panel function test
G). Enter number “7”: for touch panel calibration (5 point)
H). Enter number “8”: for LCD function test
I). Enter number “9”: for backlight brightness adjustment function test

A). Enter number “1”: for I²C EEPROM function test (1st 2nd 3rd EEPROM)

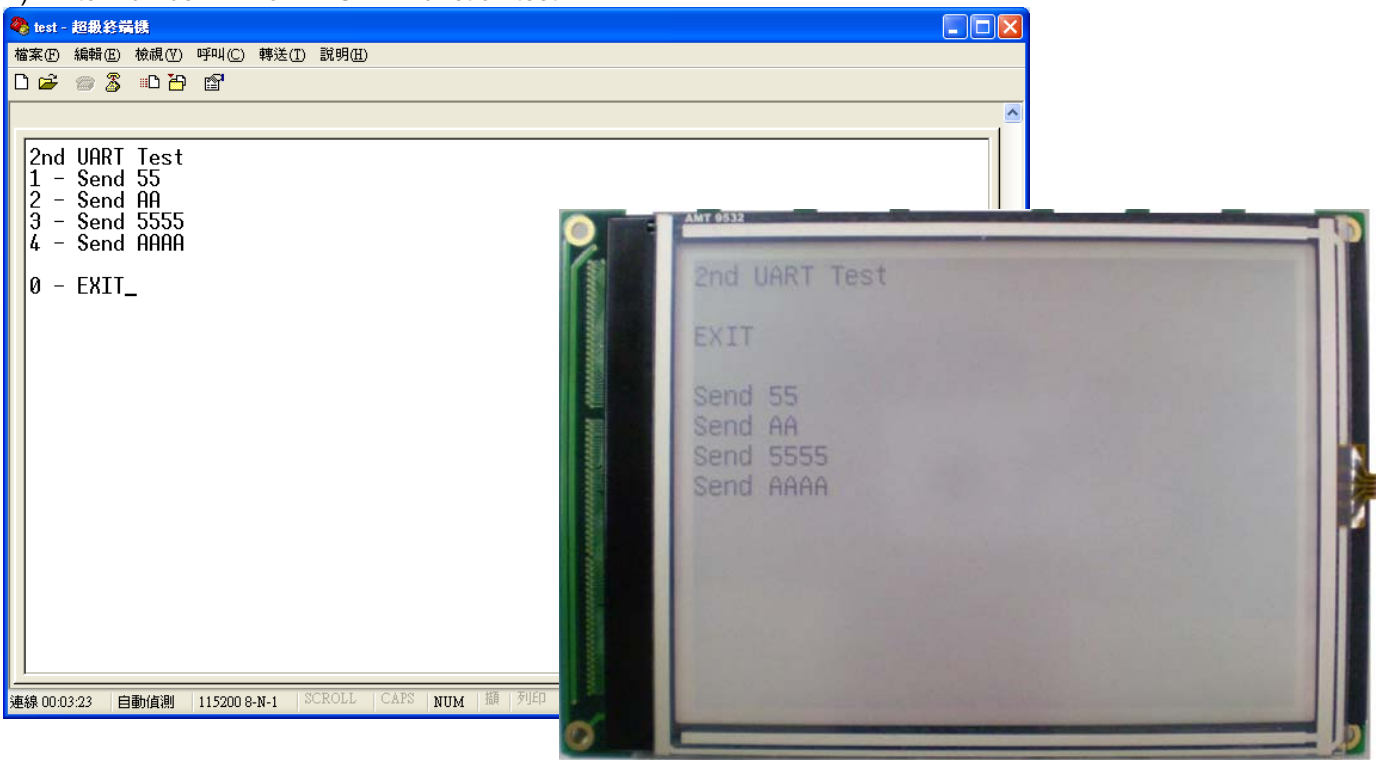


1). Enter number “1”: for I²C EEPROM function test (1st EEPROM)
2). Enter number “2”: for I²C EEPROM function test (2nd EEPROM)
3). Enter number “3”: for I²C EEPROM function test (3rd EEPROM)
4). Enter number “0”: to return to main screen



1). Enter number “1”: to write 55 on current screen
2). Enter number “2”: to write AA on current screen
3). Enter number “3”: to write 00 on current screen
4). Enter number “4”: to write FF on current screen
5). Enter “P”: to switch to previous page
6). Enter “N”: to switch to next page
7). Enter number “0”: to return to the previous page

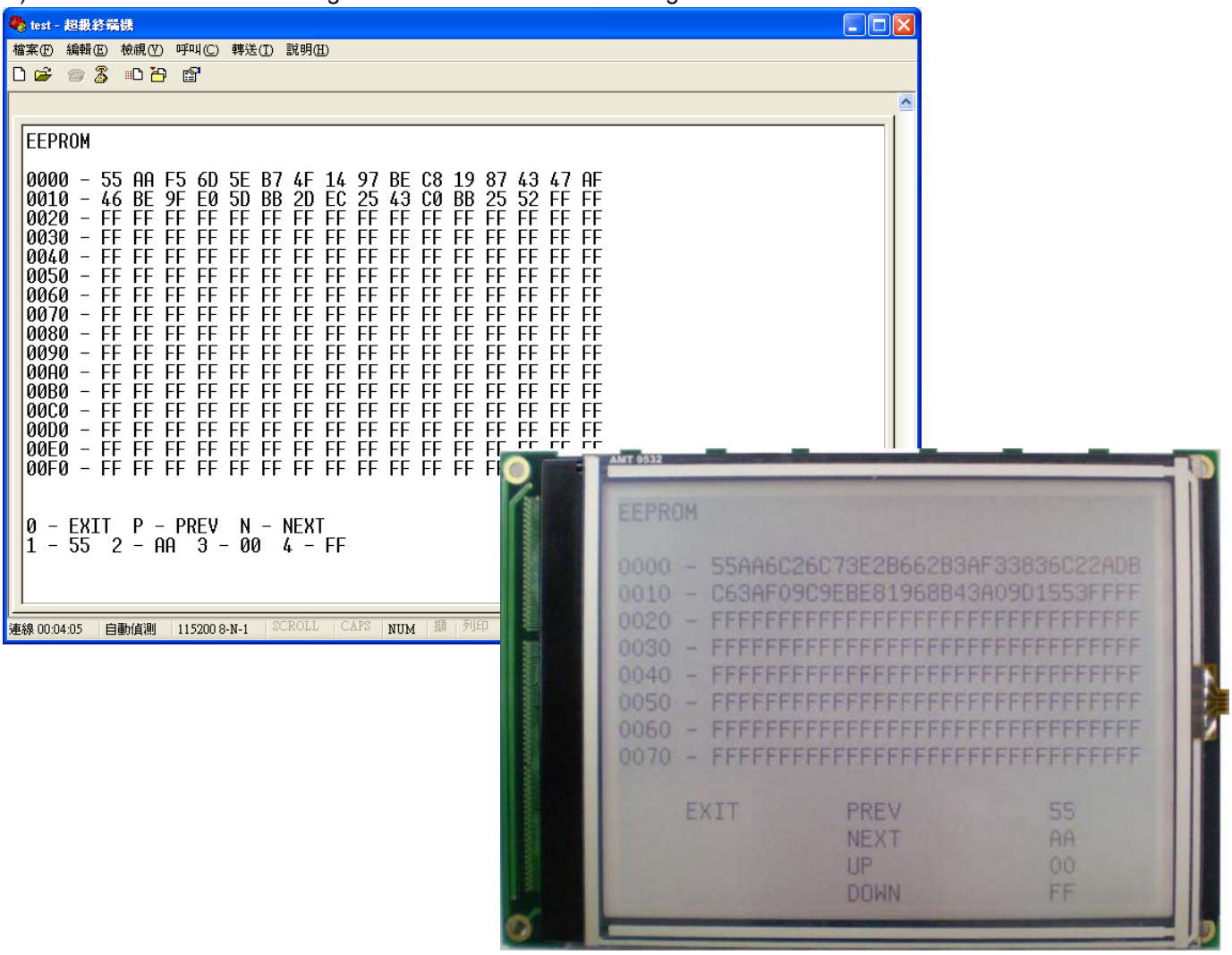
B). Enter number “2”: for 2nd UART function test



1). Enter number “1”: to send 1 byte (0x55)
2). Enter number “2”: to send 1 byte (0xAA)
3). Enter number “3”: to send 1 word (0x5555)
4). Enter number “4”: to send 1 byte (0xAAAA)
5). Enter number “0”: to return to main screen

Auto receive byte or word

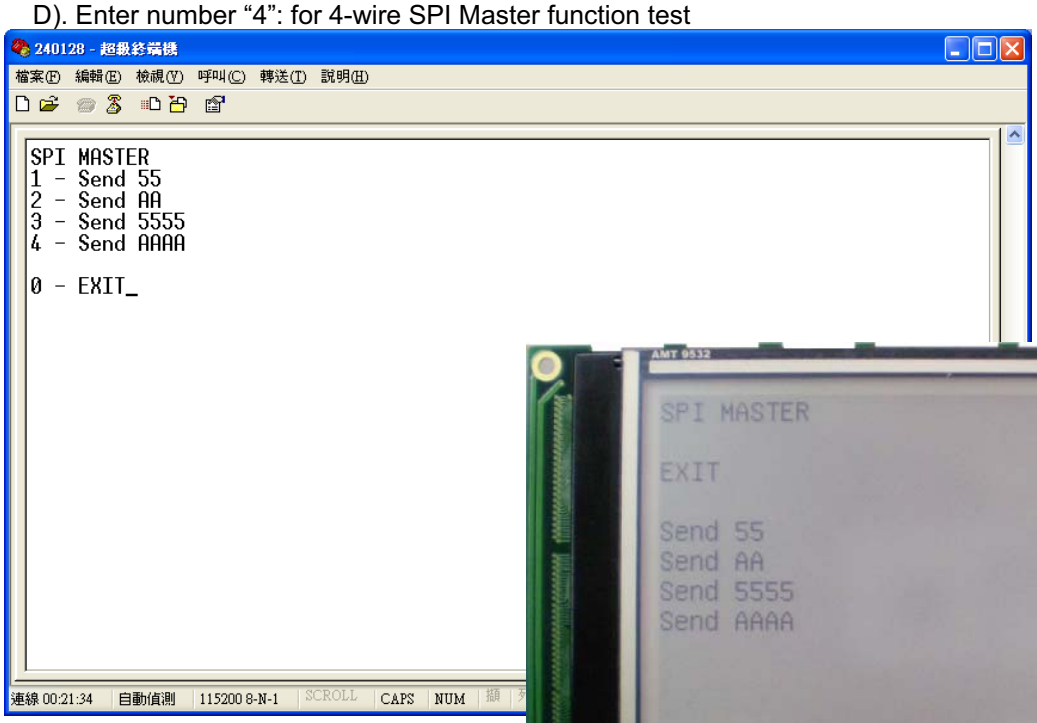
C). Enter number “3”: for ATmega644P internal EEPROM setting



** Address 0x0000~0x001D is for touch panel calibration data

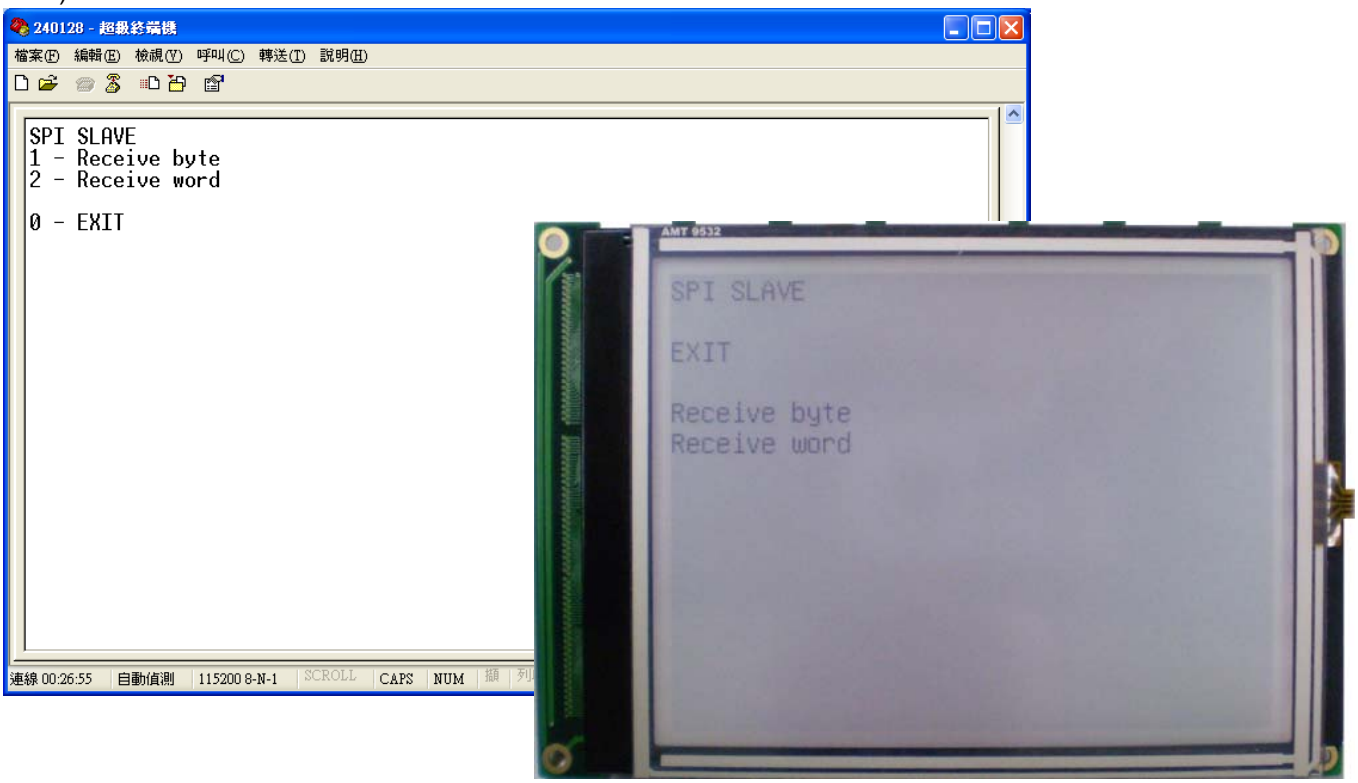
1). Enter number “1”: to write 55 on current screen
2). Enter number “2”: to write AA on current screen
3). Enter number “3”: to write 00 on current screen
4). Enter number “4”: to write FF on current screen
5). Enter “P”: to switch to previous page
6). Enter “N”: to switch to next page
7). Enter number “0”: to return to main screen

D). Enter number “4”: for 4-wire SPI Master function test



1). Enter number “1”: to send 1 byte (0x55)
2). Enter number “2”: to send 1 byte (0xAA)
3). Enter number “3”: to send 1 word (0x5555)
4). Enter number “4”: to send 1 byte (0xAAAA)
5). Enter number “0”: to return to main screen

E). Enter number “5”: for 4-wire SPI Slave function test

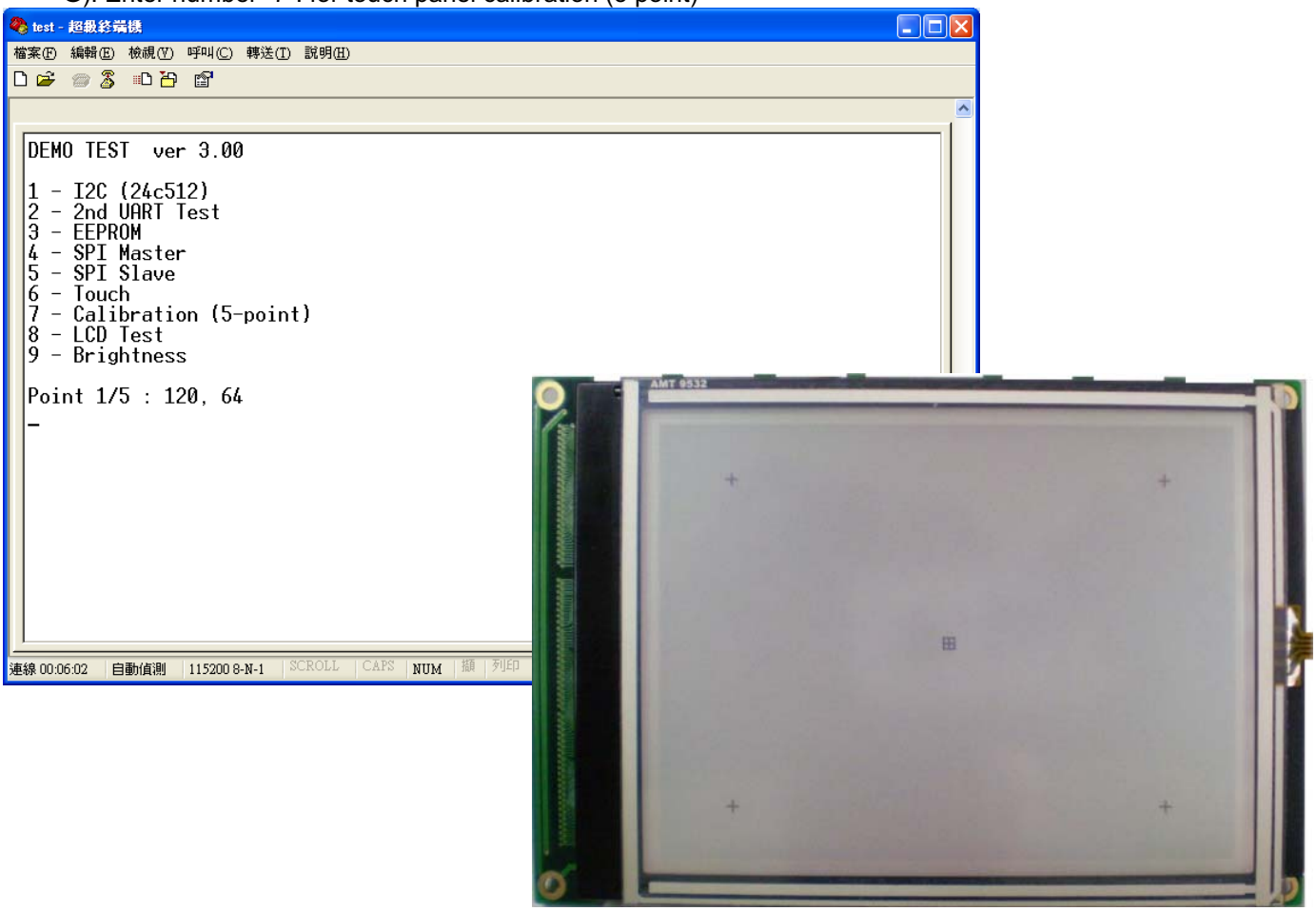


1). Enter number “1”: to receive 1 byte
2). Enter number “2”: to receive 1 word
3). Enter number “0”: to return to main screen

F). Enter number “6”: to read from touch panel

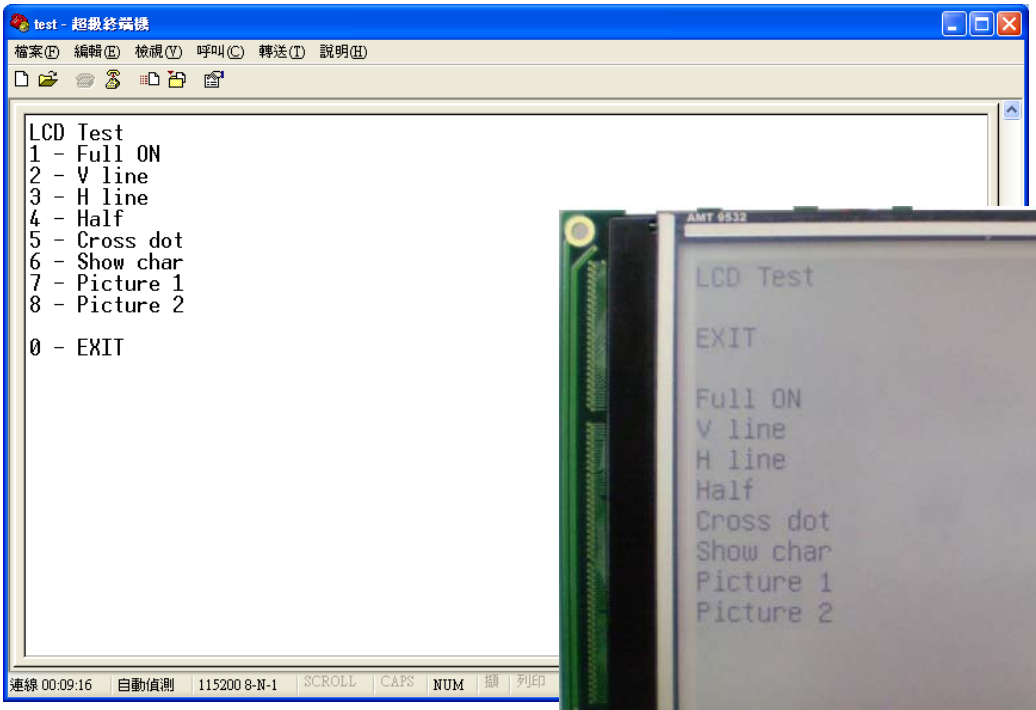
1). Using light pen to touch a point on touch panel
2). X, Y values are touch panel coordinates
3). DX, DY values are LCD coordinates (DX=0/DY=0 if touch panel is not calibrated.)
4). Enter number “0”: to return to main screen

G). Enter number “7”: for touch panel calibration (5 point)



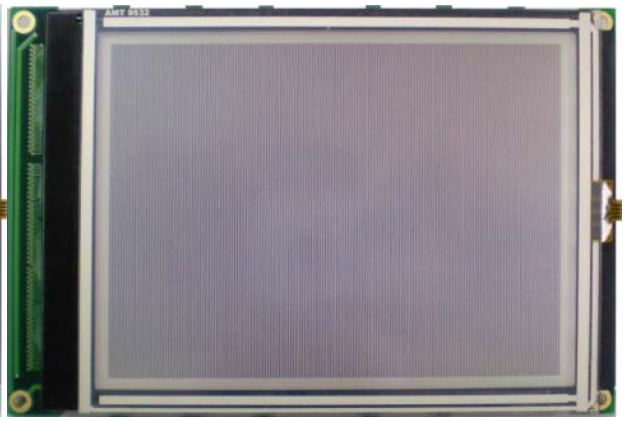
1). On the LCD, designer will see ⊞ displays. Please use light pen to touch center point of until it turns to ⊞ be ⛶. Such, one point is calibrated.
2). Please repeat above process to calibrate 5 points.
3). Touch panel calibration is finished.

H). Enter number “8”: for LCD function test

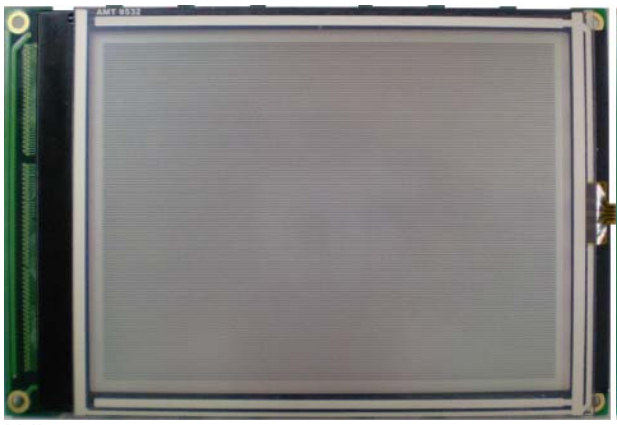


1). Enter number “1”: to LCD show full on
2). Enter number “2”: to LCD show vertical line
3). Enter number “3”: to LCD show horizontal line
4). Enter number “4”: to LCD show half
5). Enter number “5”: to LCD show cross dot
6). Enter number “6”: to LCD show character
7). Enter number “7”: to LCD show picture 1
8). Enter number “8”: to LCD show picture 2
9). Enter number “0”: to return to main screen

Full on

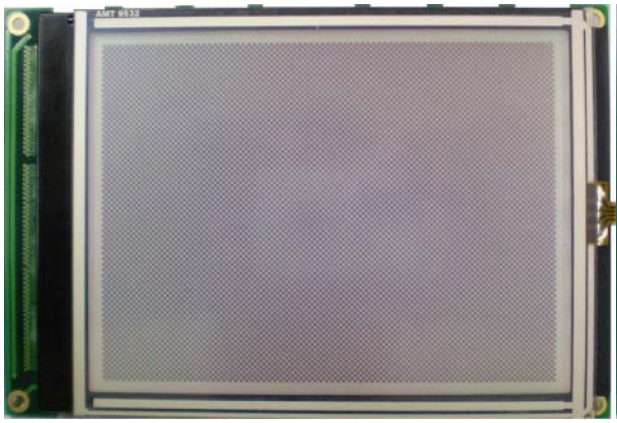


V line



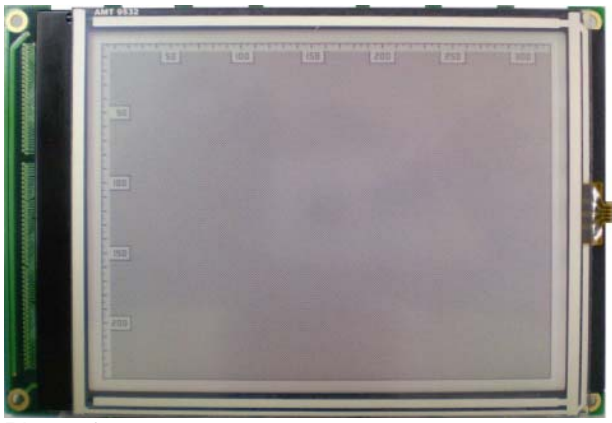
H line

Half



Cross dot

Character



Picture1

Picture2

l). Enter number “9”: for backlight brightness adjustment function test

1). Enter number “1”: to backlight brightness + 5
2). Enter number “2”: to backlight brightness - 5
3). Enter number “0”: to return to main screen

4-4 Bolymin Free Software Utilities

4-4-1 Website Links

- Touch Panel Driver

www.bolymin.com.tw/embedded/utility/BEGV643A/tpdriver.zip

- UART Driver (RS-232/485/422)

www.bolymin.com.tw/embedded/utility/BEGV643A/uartdriver.zip

- SPI Driver

www.bolymin.com.tw/embedded/utility/BEGV643A/spidriver.zip

- I^2C Driver

www.bolymin.com.tw/embedded/utility/BEGV643A/i2cdriver.zip

- Backlight Driver

www.bolymin.com.tw/embedded/utility/BEGV643A/backlightdriver.zip

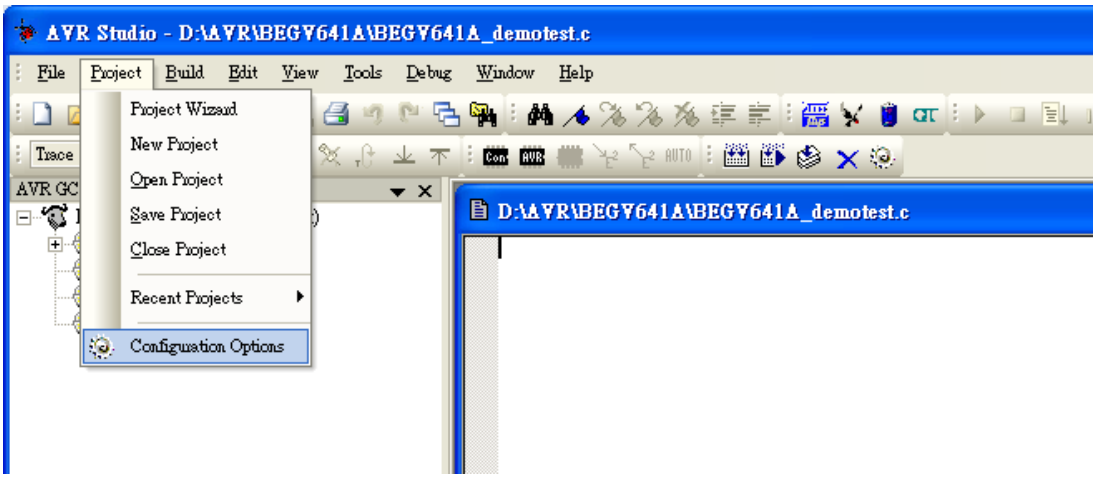
- LCD Driver

www.bolymin.com.tw/embedded/utility/BEGV643A/lcddriver.zip

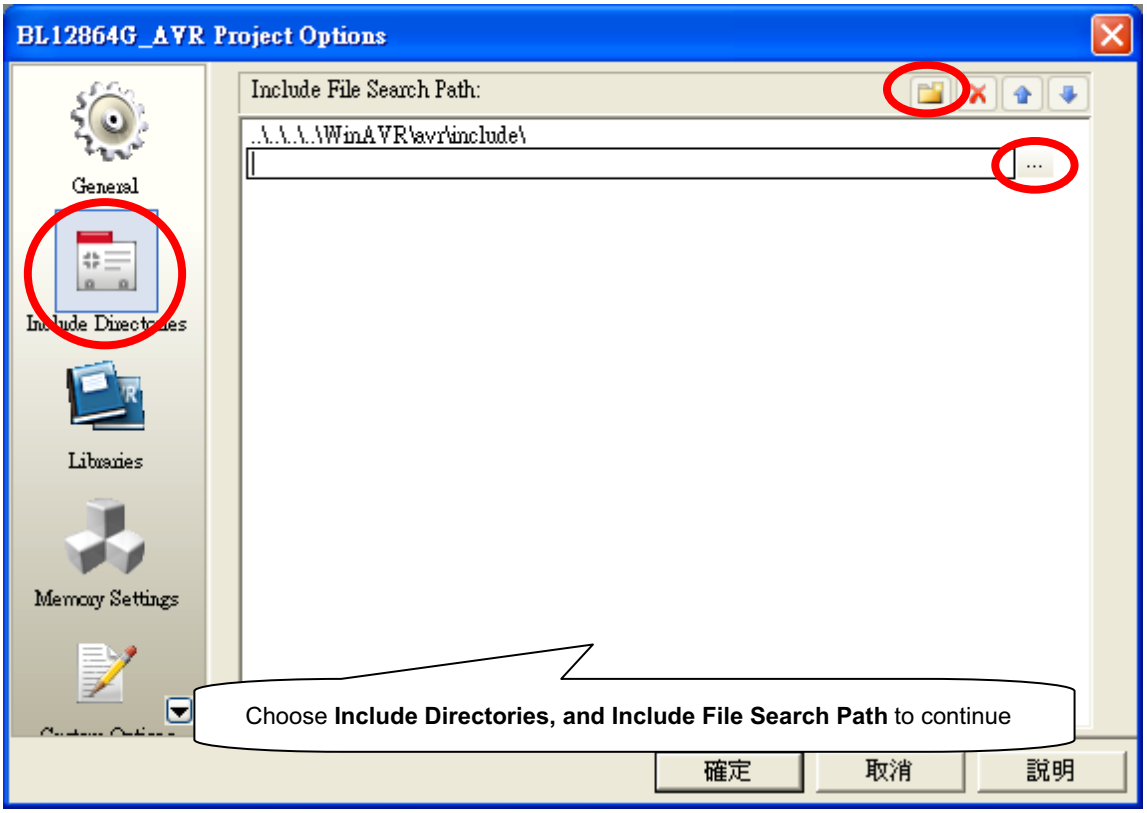
4-4-2 Introduction of Bolymin software utilities

It is recommended to use Bolymin software utilities in order to speed up project development phase. However, designer may develop your own software utilities, if you find Bolymin utilities is not convenient to use.

On following paragraphs, we explain the way to add Bolymin software utilities into designer PC.

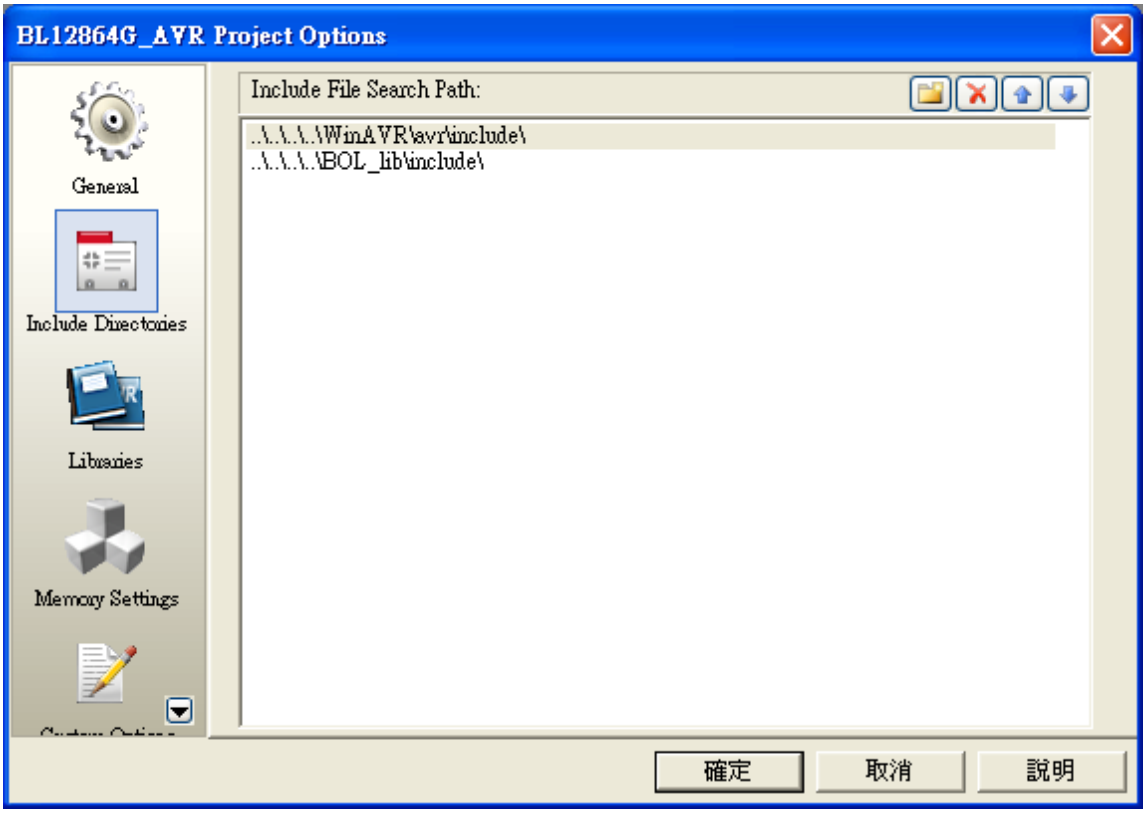


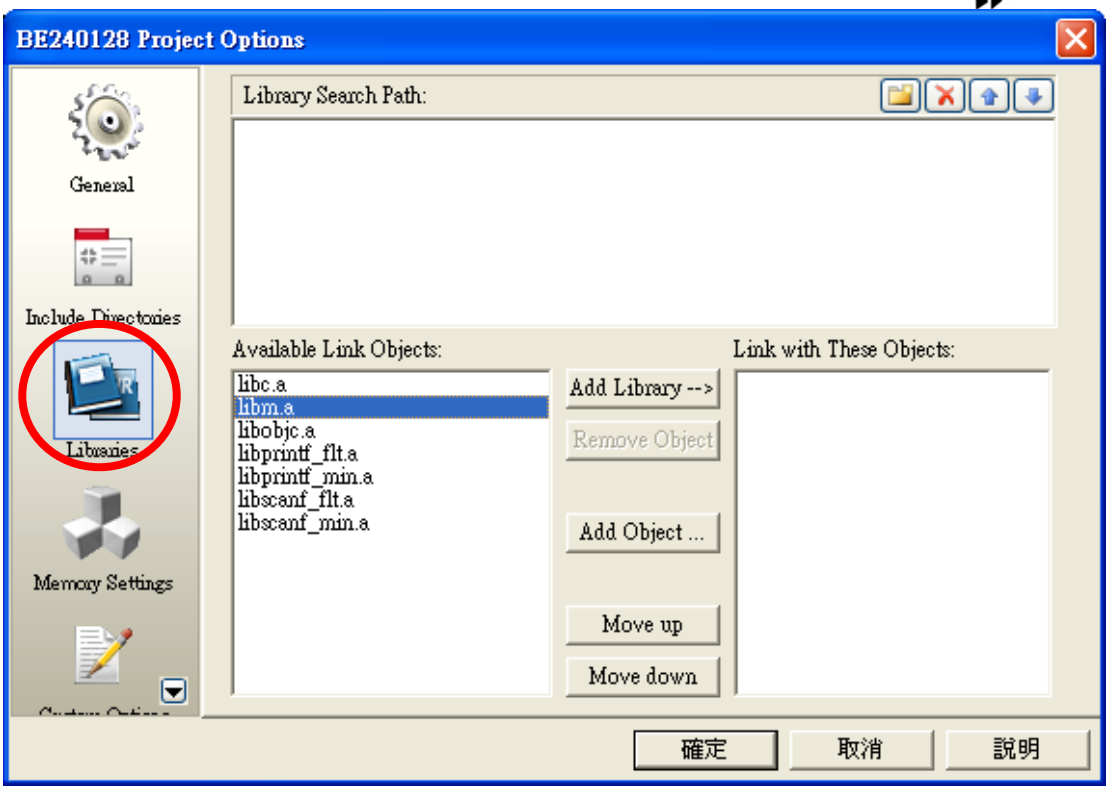
Note: Under AVR Studio4, Project→Configuration Options



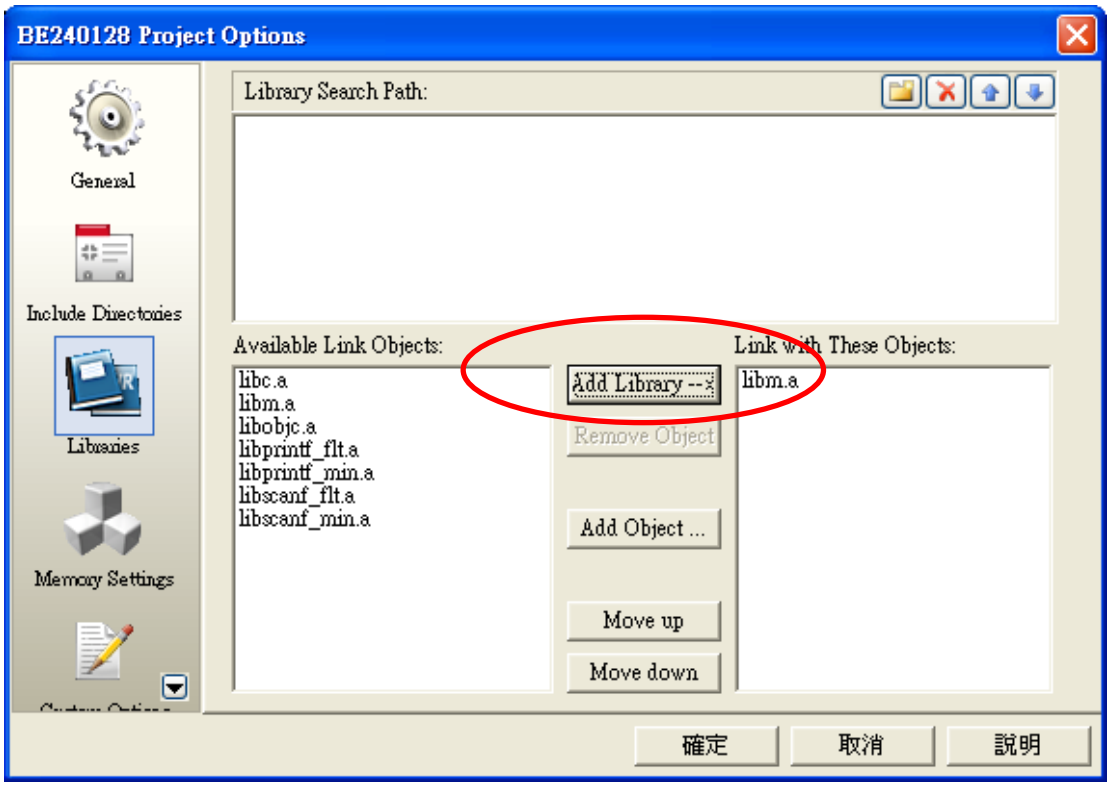
Note: Adding into **Header files**

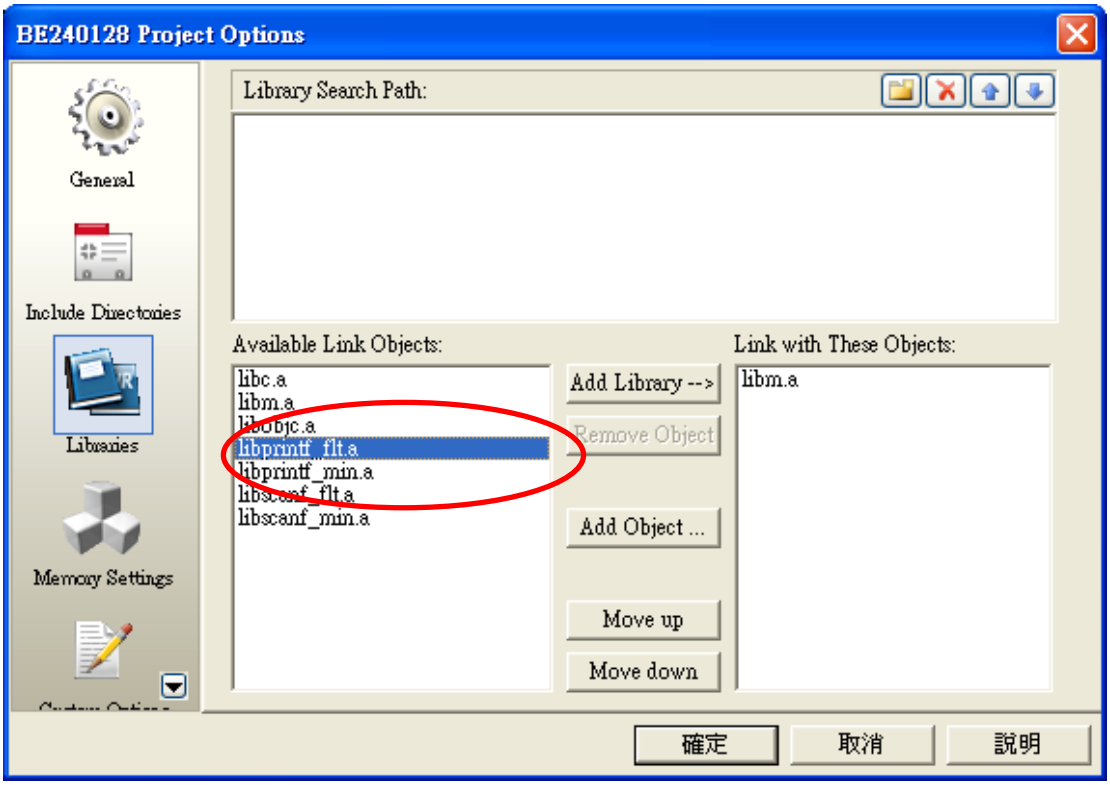
Choose **Include** as File Search Path to continue

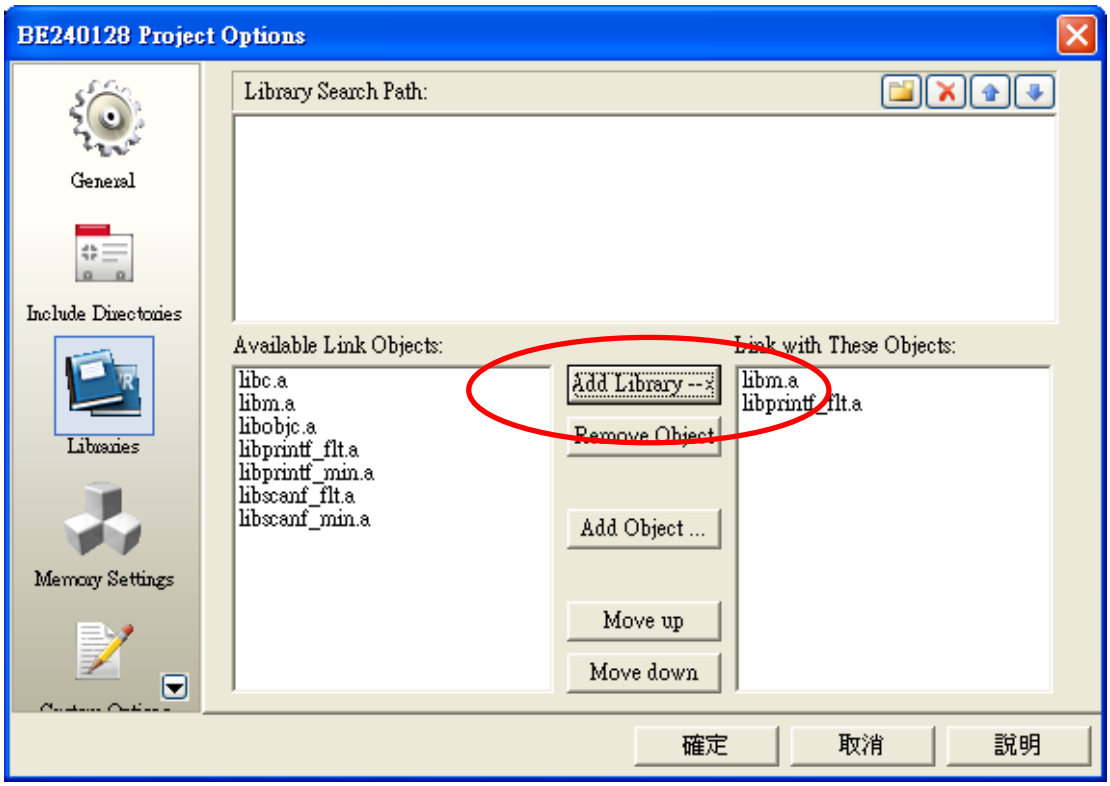


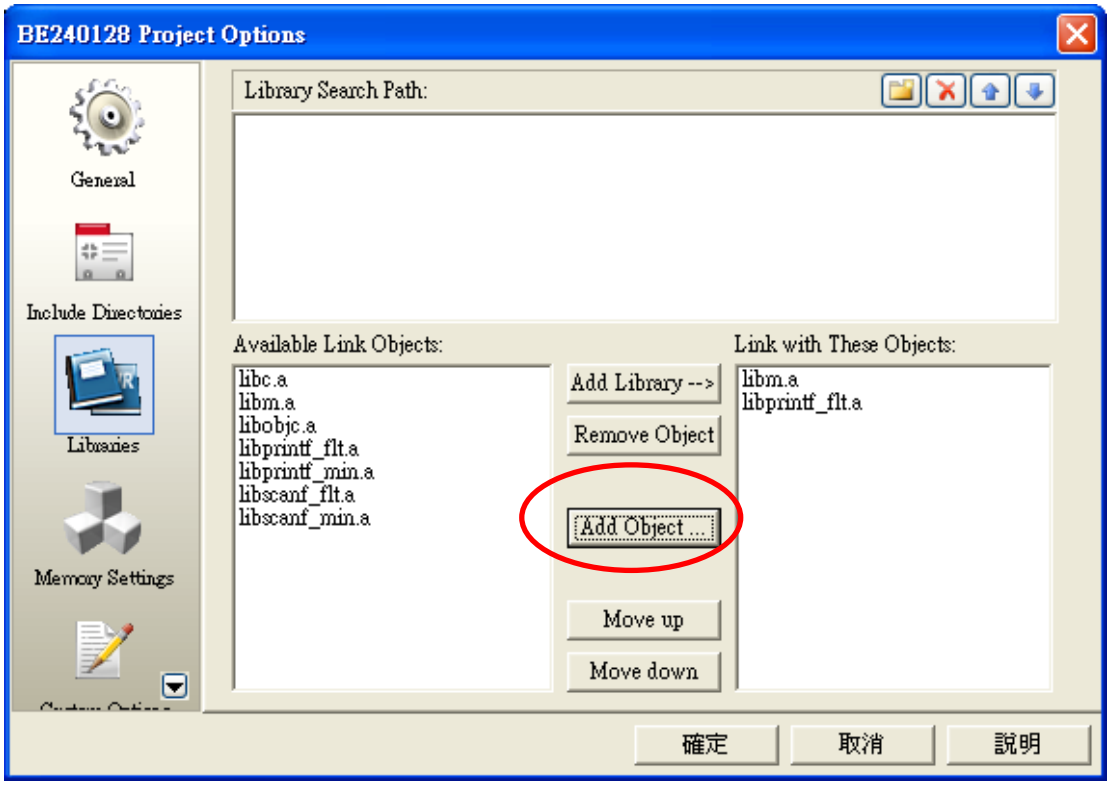


Note: For touch panel calibration, Floating-point operations is necessary, so please add libm.a and libprintf_flt.a in WinAVR.

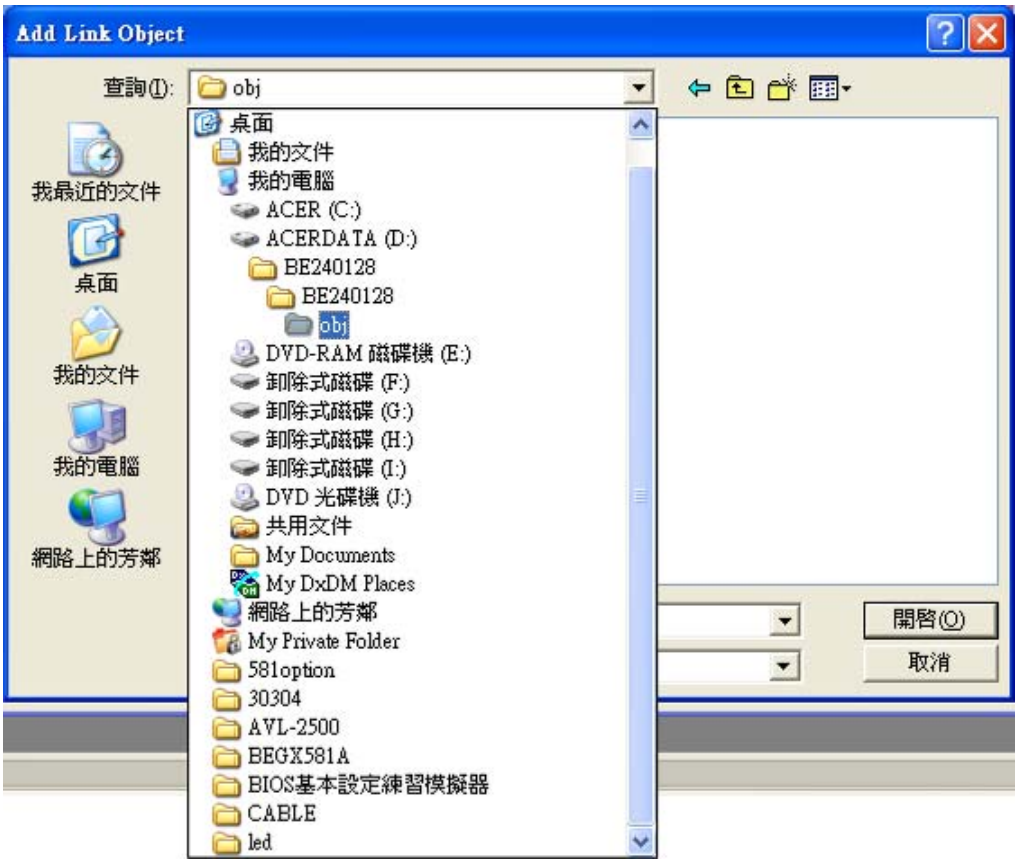


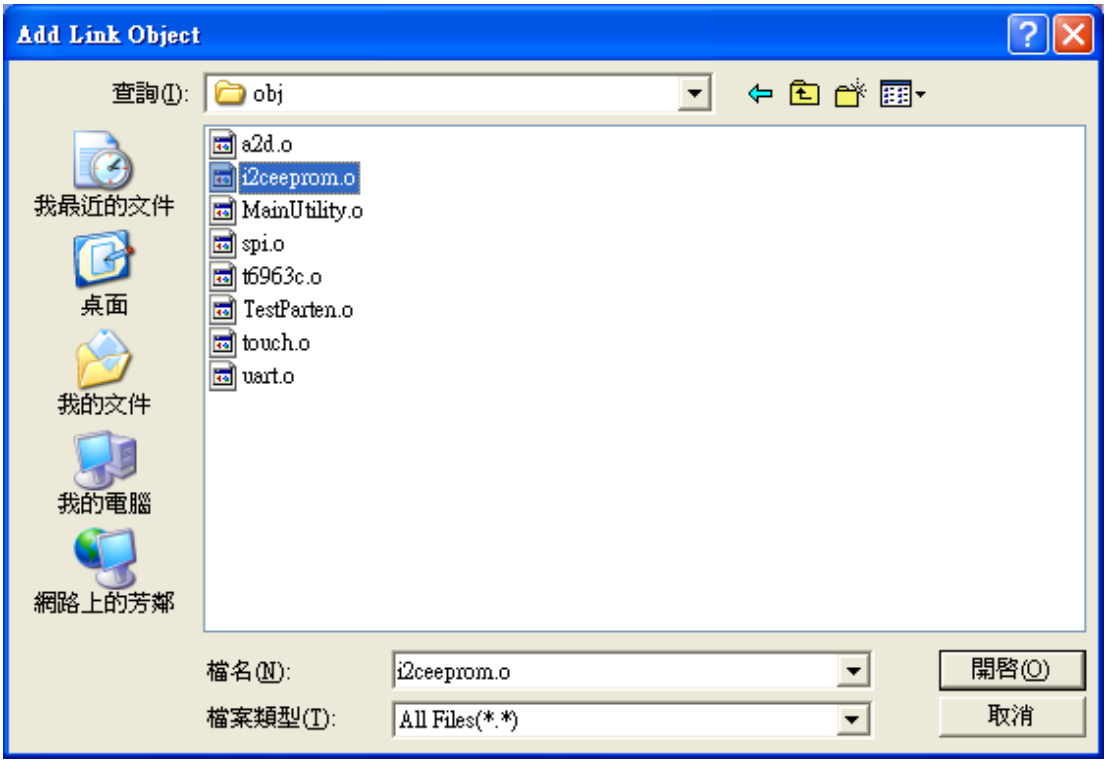

BE240128 Project Options
General
Include Directories
Libraries
Memory Settings
Library Search Path:
Available Link Objects:
libc.a
libm.a
libobjc.a
libprintf_flt.a
libprintf_min.a
libscanf_flt.a
libscanf_min.a
Add Library -->
Remove Object
Add Object ...
Move up
Move down
Link with These Objects:
libm.a
確定
取消
說明

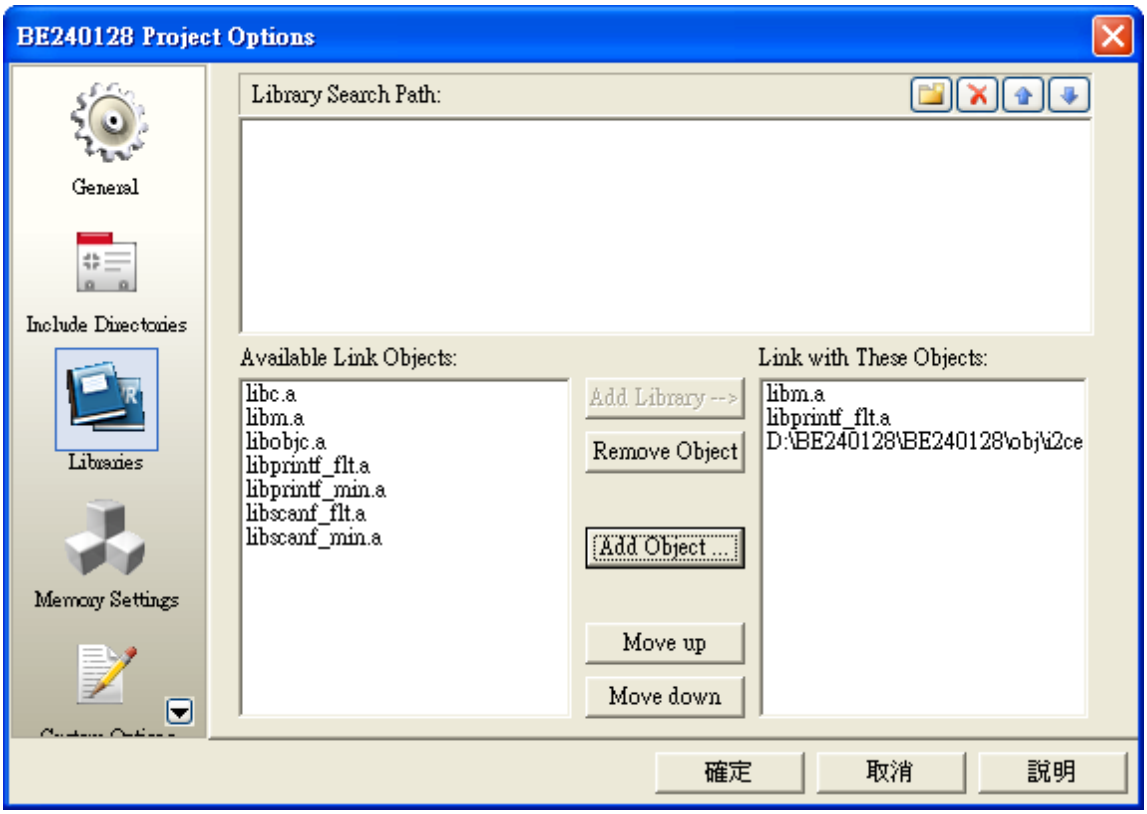

BE240128 Project Options
General
Include Directories
Libraries
Memory Settings
Library Search Path:
Available Link Objects:
libc.a
libm.a
libobjc.a
libprintf_flt.a
libprintf_min.a
libscanf_flt.a
libscanf_min.a
Add Library -->
Remove Object
Add Object ...
Move up
Move down
Link with These Objects:
libm.a
libprintf_flt.a
確定
取消
說明

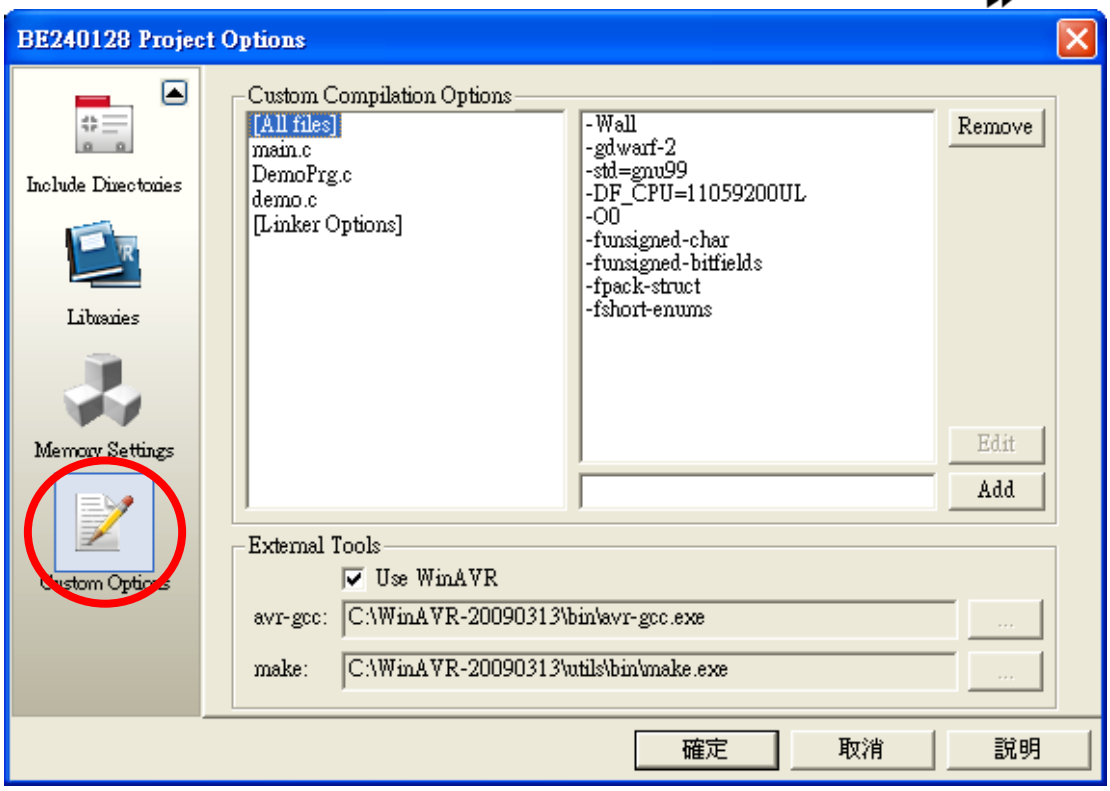


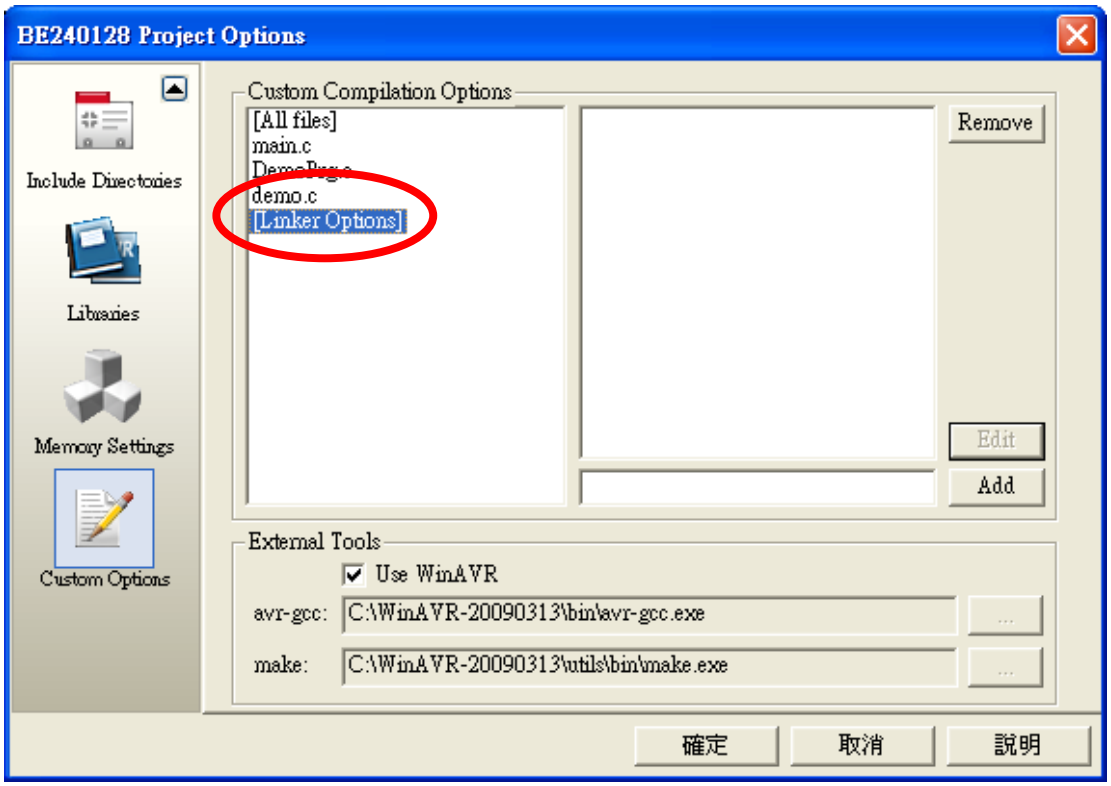
Note: Adding into Obj



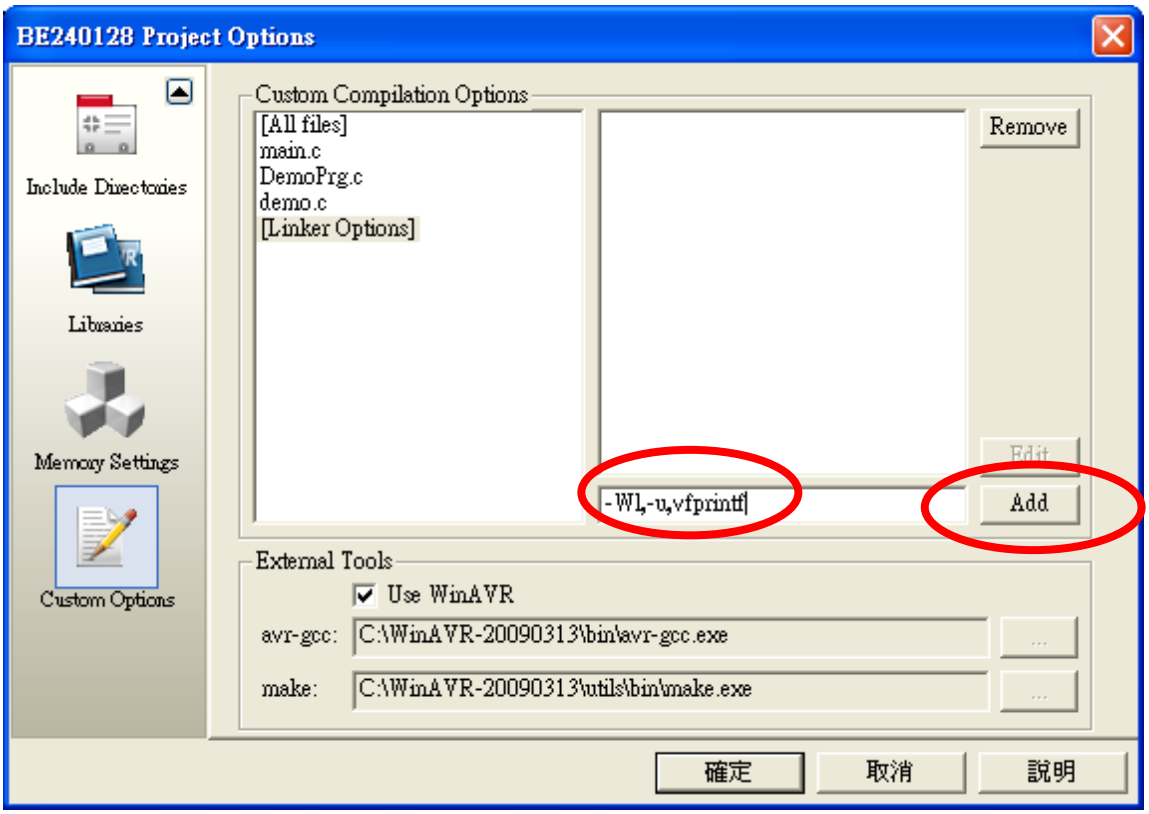

Add Link Object
查詢(I):
obj
a2d.o
i2ceeprom.o
MainUtility.o
spi.o
t6963c.o
TestParten.o
touch.o
uart.o
我最近的文件
桌面
我的文件
我的電腦
網路上的芳鄰
檔名(N):
i2ceeprom.o
檔案類型(T):
All Files(*.*)
開啓(O)
取消


BE240128 Project Options
General
Include Directories
Libraries
Memory Settings
Library Search Path:
Available Link Objects:
libc.a
libm.a
libobjc.a
libprintf_flt.a
libprintf_min.a
libscanf_flt.a
libscanf_min.a
Add Library -->
Remove Object
Add Object ...
Move up
Move down
Link with These Objects:
libm.a
libprintf_flt.a
D:\BE240128\BE240128\obj\i2ce
確定
取消
說明

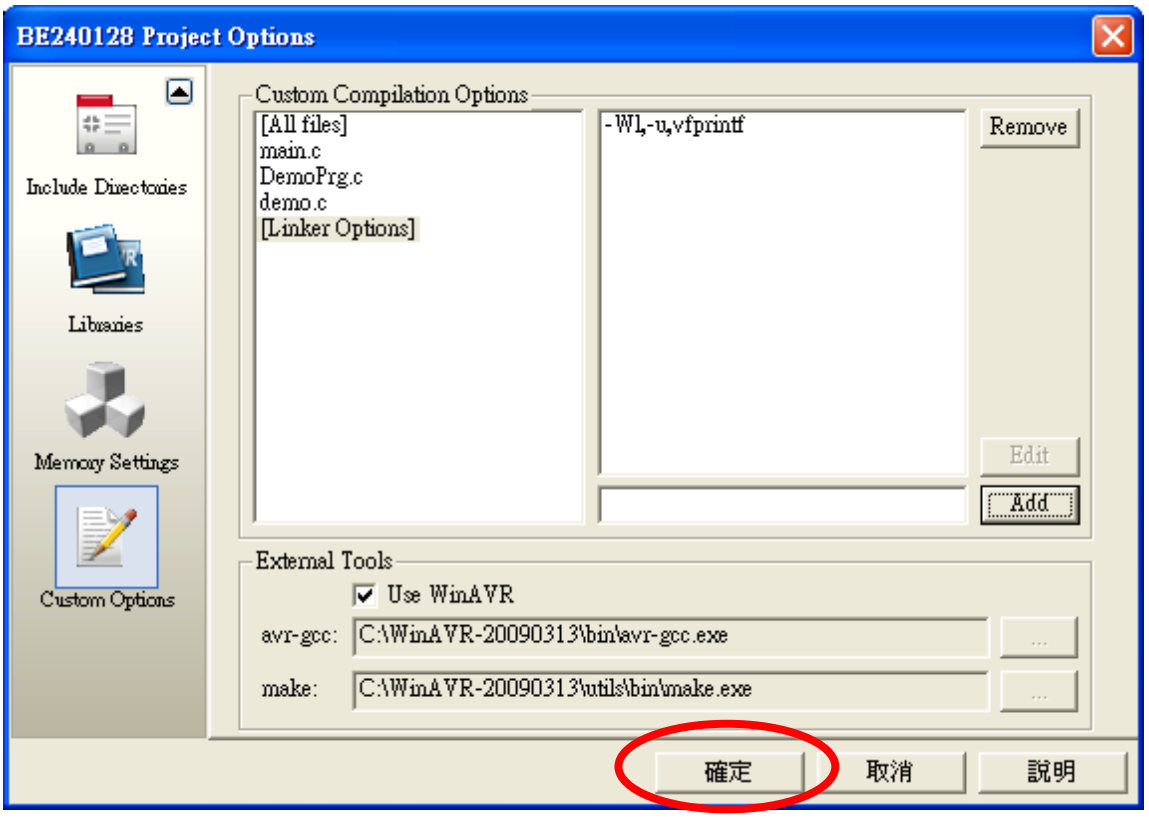




Note: choose [Linker Options]



Note: key in -Wl,-u,vfprintf, and then **Add**



4-4-3 Software Utilities Function Description

4-4-3-1 UART function

Header file : uart.h
Object file : uart.o

uartInit Function: Initial UART.

Syntax	void uartInit(uint8_t byPort, uint32_t uBaudrate, uint8_t byParity, uint8_t uDatabit, uint8_t uStopbit, uint8_t nTxMode);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port nBaudrat Baud rate, ex: 9600. byParity Parity Check, 'N' – None, 'E' – EVEN, 'O' - ODD. uDatabit data bit , 5 ~ 8. uStopbit data bit , 1 ~ 2. nTxMode Transmission Mode. 0 or FALSE – RS232. 1 or TRUE – RS485 or RS422.
Return value	None.

uartSetBaudRate Function: to set up baud rate for assigned UART port

Syntax	void uartSetBaudRate(uint8_t byPort, uint32_t uBaudrate);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port nBaudrate Baud Rate, ex: 9600.
Return value	None.

uartSendByte Function: To send 1 byte from assigned UART port

Syntax	void uartSendByte(uint8_t byPort, uint8_t txData);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port txData byte to be sent.
Return value	None.

uartDisablePort Function: to stop operation of assigned Uart port

Syntax	void uartDisablePort (uint8_t byPort);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port
Return value	None.

uartSendString Function: to send 1 string from assigned UART port

Syntax	void uartSendString(uint8_t byPort uint8_t* str);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port str Index of string to be sent, ending with “0”.
Return value	None.

uartSendBuffer Function: to send buffer from assigned UART port

Syntax	void uartSendBuffer(uint8_t byPort, uint8_t* buffer, uint16_t nBytes);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port buffer index of buffer to be sent nBytes bytes of buffer to be sent
Return value	None.

uartReceiveByte Function: to read 1 byte data from assigned UART port

Syntax	uint8_t uartReceiveByte(uint8_t byPort, uint8_t* rxData);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port rxData index of data to be received
Return value	TRUE – rxData is true data FALSE – UART port no data

uartReceiveBufferIsEmpty Function: to check if there is data in assigned UART port

Syntax	void uartReceiveBufferIsEmpty(uint8_t byPort);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port
Return value	TRUE – There is data on assigned UART port FALSE – There is no data on assigned UART port

uartFlushReceiveBuffer Function: to clear receiving buffer on assigned UART port

Syntax	void uartFlushReceiveBuffer(uint8_t byPort);
Parameters	byPort UART_PORT0 – 1st Uart port UART_PORT1 – 2nd Uart port
Return value	None.

uartEnableTx Function: to Enable or Disable UART transmitter. (When UART port is applied on RS485 or RS422 , transmitter must be set to Disable, and to Enable transmitter only when sending data.)

Syntax	void uartEnableTx(uint8_t bEnable);
Parameters	bEnable TRUE – Enable transmitter FALSE – Disable transmitter.
Return value	None.

4-4-3-2 I^2C function

Header file : i2ceeprom.h
object file : i2ceeprom.o

i2cInitial Function: Initial I2C functions. User should call this function before using I2C functions.

Syntax	void i2cInitial( );</br>
Parameters	None.
Return value	None.

i2cReadByte Function: to read 1 byte data from I^2C

Syntax	uint8_t i2cReadByte(uint8_t uDevAddr, uint16_t nAddr);</br></br></br>
Parameters	uDevAddr I^2C device address. (address of three 24c512 on board are: $A2_{hex}$, $A4_{hex}$, $A6_{hex}$. nAddr address to write in .</br>
Return value	Data reading from I^2C

i2cWriteByte Function: to write 1 byte data from I^2C

Syntax	void i2cWriteByte(uint8_t uDevAddr, uint16_t nAddr, uint8_t byData);</br></br></br>
Parameters	uDevAddr I^2C device address.(address of three 24c512 on board are: $A2_{hex}$, $A4_{hex}$,$A6_{hex}$. nAddr address to write in . byData data to write in .</br></br>
Return value	None.

i2cSetSpeed Function: to set baud rate for I2C

Syntax	void i2cSetSpeed(uint16_t nSpeed,);</br></br>
Parameters	uSpeed I^2C baud rate , 0 – 100K, 1 – 250K.
Return value	None.

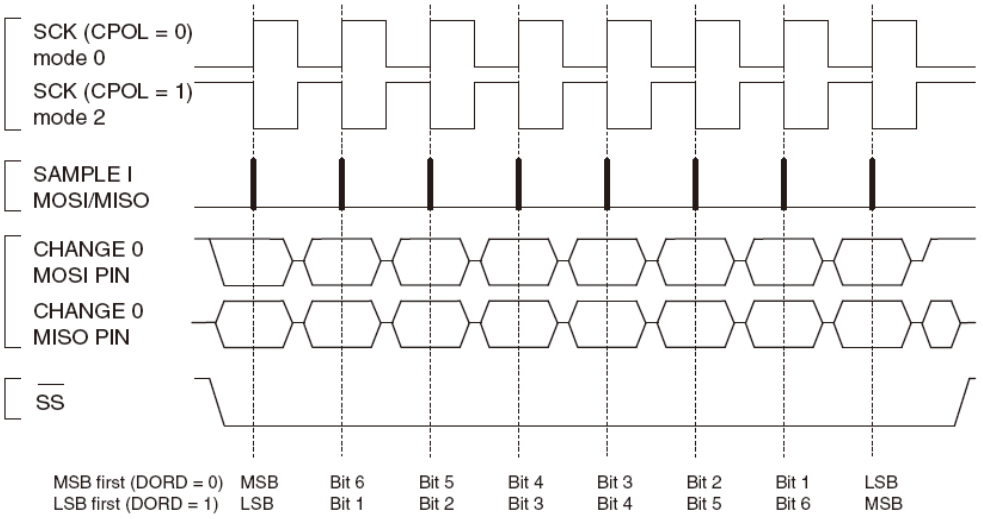
4-3-3-3 SPI function

Header file : spi.h
object file : spi.o

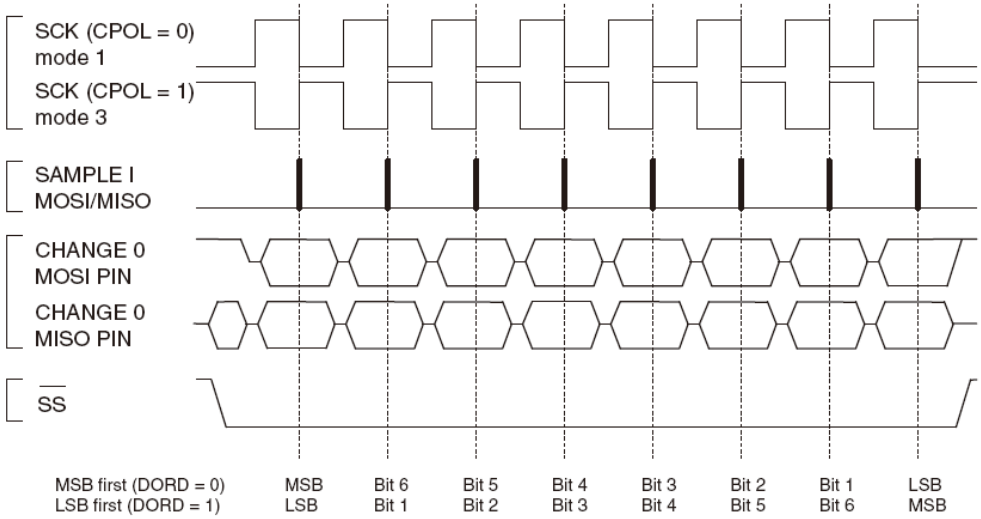
spiMaster Function: Initial SPI to master mode.

Syntax	void spiMaster(uint8_t mode);
Parameters	mode SPI mode. 0 – CPOL=0, CPHA=0 Sample (Rising) Setup (Falling) 1 – CPOL=0, CPHA=1 Setup (Rising) Sample (Falling) 2 – CPOL=1, CPHA=0 Sample (Falling) Setup (Rising) 3 – CPOL=1, CPHA=1 Setup (Falling) Sample (Rising)
Return value	None.

SPI Transfer Format with CPHA = 0



SPI Transfer Format with CPHA = 1



spiSlave Function: Initial SPI to slave mode.

Syntax	void spiSlave(uint8_t mode,);
Parameters	mode SPI mode. 0 – CPOL=0, CPHA=0 Sample (Rising) Setup (Falling) 1 – CPOL=0, CPHA=1 Setup (Rising) Sample (Falling) 2 – CPOL=1, CPHA=0 Sample (Falling) Setup (Rising) 3 – CPOL=1, CPHA=1 Setup (Falling) Sample (Rising)
Return value	None.

spiSetBitrate Function: to set SPI baud rate

Syntax	void spiSetBitrate(uint8_t spr);
Parameters	spr SPI baud rate. system OSC is 11.0592MHz. 0 – OSC / 4. 1 – OSC / 16. 2 – OSC / 64. 3 – OSC / 128.
Return value	None.

spiSetDataOrder Function: to set SPI order when sending data, LSB first or MSB first.

Syntax	void spiSetDataOrder(uint8_t order);
Parameters	order LSB first or MSB first. DATA_LSB_FIRST–LSB of the data word will be transmitted first. DATA_MSB_FIRST–MSB of the data word will be transmitted first.(Default)
Return value	None.

spiSendByte Function: to send 1 byte data from SPI port

Syntax	void spiSendByte(uint8_t data);
Parameters	data byte to be sent
Return value	None.

spiRecvByte Function: to receive 1 byte data from SPI port

Syntax	uint8_t spiRecvByte( );
Parameters	None.
Return value	To read 1 byte data from SPI

spiTransferByte Function: to send and read 1 byte data from SPI port

Syntax	uint8_t spiTransferByte(uint8_t data);
Parameters	data byte to be sent
Return value	1 byte data reading from SPI

spiTransferWord Function: to send and read 1 word from SPI port

Syntax	Uint16_t spiTransferByte(uint16_t data);
Parameters	data word to be sent
Return value	1 word data reading from SPI

4-3-3-4 E^2PROM function

E^2PROM Function is built inside WinAVR, so users need only to include eeprom.h in program to call E^2PROM Function.

For exmaple：

```
#include <avr/eeprom.h>
```

eeprom_write_byte Function: to write 1 byte to MCU EEPROM.

Syntax	void eeprom_write_byte (uint8_t* address, uint8_t value);
Parameters	address E^2PROM address to write in, range 0 ~ 0x7FF value E^2PROM data to write in
Return value	None.

eeprom_read_byte Function: This function read one byte from EEPROM.

Syntax	uint8_t eeprom_read_byte (uint8_t* address);
Parameters	address E^2PROM address to read, range 0 ~ 0x7FF
Return value	Data reading from E^2PROM

eeprom_write_block Function: to write block data to EEPROM.

Syntax	void eeprom_write_byte (void* pointer_ram, const void* pointer_eeprom, size_t n);
Parameters	pointer_ram index of block data to write in . pointer_eeprom E^2PROM address to write in, range 0 ~ 0x7FF. n length of E^2PROM data to write in
Return value	None.

eeprom_read_block Function: to read block data from EEPROM

Syntax	uint8_t eeprom_read_block (void* pointer_ram, const void* pointer_eeprom, size_t n);
Parameters	pointer_ram index of block data to read . pointer_eeprom address of E^2PROM to read, rnage 0 ~ 0x7FF. n length of E^2PROM data to read
Return value	Data block read to .

4-3-3-5 Touch function

Header file : touch.h, a2d.h
object file : touch.o, a2d.o

touchInit Function: Initial Touch panel.

Syntax	void touchInit( );
Parameters	None.
Return value	None.

touchGet Function: to read touch data from touch panel

Syntax	uint8_t touchGet(int * pX, int * pY);
Parameters	pX to read X Coordinate from touch data pY to read Y Coordinate from touch data
Return value	TRUE data of pX and pY is true FALSE data of pX and pY is false

touchDrawCalPoint Function: to draw Calibration cross Coordinate on LCD

Syntax	uint8_t touchDrawCalPoint (POINT* ptCal, int n);
Parameters	ptCal Calibration Coordinate. n Calibration Coordinate No.
Return value	None.

setCalibrationMatrix Function: to set Calibration calculation matrix

Syntax	void setCalibrationMatrix(POINT * ptDisplay, POINT * ptTouch, int n);
Parameters	ptDisplay LCD reference Coordinate for calibration. ptTouch Touch Coordinate for calibration n Coordinate No. for calibration
Return value	None.

getDisplayPoint Function: to change Touch Coordinate into LCD

Syntax	void getDisplayPoint(int x, int y, int * pX, int * pY);
Parameters	x Touch X Coordinate. y Touch Y Coordinate. pX LCD X Coordinate changed from Touch X Coordinate pY LCD Y Coordinate changed from Touch Y Coordinate
Return value	None.

4-3-3-6 LCD control function

Header file : RA8806.h
object file : RA8806.o

lcdInit Function: Initialize all parameters of LCD display. User should call this function before use functions of LCD display.

Syntax	void lcdInit ( );
Parameters	None.
Return value	None.

lcdDisplayClr Function: Clear screen (include graphic and text layer).

Syntax	lcdDisplayClr( );
Parameters	None.
Return value	None.

lcdSetCursorPos Function: Set the position of cursor. (UNIT=character=8*8 pixel)

Syntax	void lcdSetCursorPos(uint8_t x, uint8_t y);
Parameters	x X coordinate of cursor y Y coordinate of cursor
Return value	None.

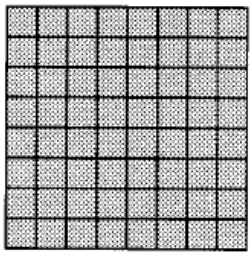
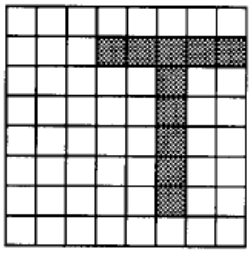
lcdSwitchDisplay Function: ON/OFF.the display of cursor, text and graphic layer.

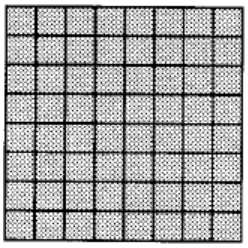
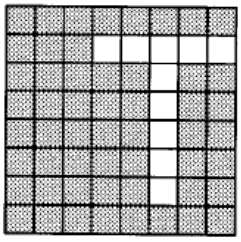
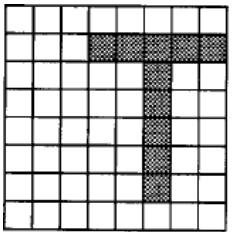
Syntax	void lcdSwitchDisplay (uint8_t display_switch);
Parameters	display_switch Display switch flag DS_DISPLAY_OFF Turn off all display. DS_TEXT_ON Turn ON the display of text layer. DS_GRAPHIC_ON Turn ON the display of graphic layer. DS_CURSOR_ON Turn ON the display of cursor. DS_BLINK_ON Turn ON cursor blink. **This flag is available while cursor is ON.**
Return value	None.

Example	// Turn OFF all display.
	 lcdSwitchDisplay(DS_DISPLAY_OFF);
	
	 // .Turn ON the display of text and graphic layer. Turn OFF cursor.
	 lcdSwitchDisplay(DS_TEXT_ON| DS_GRAPHIC_ON);
	 // .Turn ON the display of text layer and cursor without blink. Turn OFF graphic layer.
	 lcdSwitchDisplay(DS_TEXT_ON| DS_CURSOR_ON);
	
	 // .Turn ON the display of text layer, graphic layer and cursor with blink.
	 lcdSwitchDisplay(DS_TEXT_ON| DS_GRAPHIC_ON |DS_CURSOR_ON| DS_BLINK_ON);

lcdSetDispMode Function: Set the display mode between graphic layer and text layer.

Syntax	void lcdSetDispMode (uint8_t new_mode);
Parameters	New_mode New display mode between graphic layer and text layer. Value : DM_OR_MODE, DM_XOR_MODE, DM_AND_MODE
Return value	None

Content on Graphic layer	**Content on Text layer**
	

DM_OR_MODE	**DM_XOR_MODE**	**DM_AND_MODE**
		

lcdDraw Function: Draw input binary picture on specified area of graphic area.

Syntax	void lcdDraw (uint8_t x_start, uint8_t y_start, uint8_t x_end, uint8_t y_end, uint8_t * pic_data, uint8_t mode);
Parameters	x_start X coordinate of the top-left point of input picture. (UNIT=pixel) y_start Y coordinate of the top-left point of input picture. (UNIT=pixel) x_end X coordinate of the bottom-right point of input picture. (UNIT=pixel) y_end Y coordinate of the bottom-right point of input picture. (UNIT=pixel) pic_data Bit map data will be drawn. Input 0 will reverse pixels of specified area. mode DRAW_NORMAL:Draw the picture normally. DRAW_REVERSE : Reverse the picture and then draw the picture.
Return value	None.

lcdFillByte Function: Fill input byte value on specified area of graphic layer.

Syntax	void lcdFillByte (uint8_t x_start, uint8_t y_start, uint8_t x_end, uint8_t y_end, uint8_t data, uint8_t mode);
Parameters	x_start X coordinate of the top-left point of specified area. (UNIT=pixel) y_start Y coordinate of the top-left point of specified area. (UNIT=pixel) x_end X coordinate of the bottom-right point of specified area. (UNIT=pixel) y_end Y coordinate of the bottom-right point of specified area. (UNIT=pixel) pic_data Byte value will be filled. Mode DRAW_NORMAL: Fill input value normally. DRAW_REVERSE : Reverse the input value and then fill it on the specified area
Return value	None.

lcdPrintString Function: Print input string to specified location of text layer.

Syntax	void lcdPrintString (uint8_t x_start, uint8_t y_start, char * string, uint8_t str_count);
Parameters	x_start X coordinate of start location that input string will be printed. (UNIT= character=8*8 pixel) y_start Y coordinate of start location that input string will be printed. (UNIT= character=8*8 pixel) string string will be printed to LCD str_count character count of input string
Return value	None.

lcdDrawBit Function: ON/OFF the pixel on specified location of graphic layer.

Syntax	void lcdDrawBit (uint8_t x, uint8_t y, char bit_value);
Parameters	x X coordinate of the location will be drawn. (UNIT=pixel) y Y coordinate of the location will be drawn. (UNIT=pixel) bit_value 1 : ON the pixel 0 : OFF the pixel
Return value	None.

lcdDrawRect Function: Draw rectangle by single line on graphic layer.

Syntax	void lcdDrawRect (uint8_t x_start, uint8_t y_start, uint8_t x_end, uint8_t y_end,);
Parameters	x_start X coordinate of the top-left point of rectangle. (UNIT=pixel) y_start Y coordinate of the top-left point of rectangle. (UNIT=pixel) x_end X coordinate of the bottom-right point of rectangle. (UNIT=pixel) y_end Y coordinate of the bottom-right point of rectangle. (UNIT=pixel)
Return value	None

4-3-3-7 Backlight PWM control function

Header file : bklight_pwm.h
object file : bklight_pwm.o

Note:Backlight PWM control used **TIMER0** and **INTERRUPT0**

bklPWM_Init Function: Initialize all parameters of backlight PWM control function. User should call this function before use backlight PWM control functions.

Syntax	void bklPWM_Init ( );</br>
Parameters	None.
Return value	None.

bklSetBrightness Function: Set current brightness value of backlight.

Syntax	void bklSetBrightness (uint8_t brightness);</br></br>
Parameters	Brightness New brightness value 0 – OFF backlight 1~100 – Control the brightness of backlight</br></br></br>
Return value	None.

bklGetBrightness Function: Get current brightness value of backlight

Syntax	uint8_t bklGetBrightness ( );</br>
Parameters	None.
Return value	Current brightness value of backlight. (0 ~ 100)

Appendix A: LCD Controller Specification

Appendix B: EEPROM Specification

Appendix C: ATMEL ATmega644p MCU Specification

Please download this specification from following ATMEL link:
http://www.atmel.com/dyn/resources/prod_documents/doc8011.pdf

<END of BEGV643A User Manual>